OPENVMS SYSTEM MANAGEMENT GUIDE

LAWRENCE BALDWIN

Includes Disk Containing DCL System Management Automation Tools

OpenVMS System Management Guide

DIGITAL PRESS EDITORIAL BOARD

Samuel H. Fuller, Chairman
Richard W. Beane
Donald Z. Harbert
William R. Hawe
Richard J. Hollingsworth
Richard F. Lary
Alan G. Nemeth
Jean A. Proulx
Robert M. Supnik

OpenVMS System Management Guide

Lawrence L. Baldwin, Jr.

Digital Press

Boston Oxford Melbourne Singapore Toronto Munich New Delhi Tokyo

Digital Press™ is an imprint of Butterworth–Heinemann.

A member of the Reed Elsevier group

Copyright © 1996 Lawrence L. Baldwin, Jr.

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher.

All trademarks found herein are the property of their respective owners.

Recognizing the importance of preserving what has been written, Butterworth-Heinemann prints its books on acid-free paper whenever possible.

Library of Congress Cataloging-in-Publication Data
Baldwin, Lawrence, 1965—
OpenVMS system management guide / by Lawrence L. Baldwin, Jr.
p. cm.
Includes bibliographical references and index.
ISBN 1-55558-143-9
1. VAX computers. 2. VAX/VMS. I. Title.
QA76.8.V32B35 1995
005.4'449–dc20 95-14965
CIP

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

The publisher offers discounts on bulk orders of this book.
For information, please write:

Manager of Special Sales, Digital Press
Butterworth–Heinemann
313 Washington Street
Newton, MA 02158–1626

Order number: EY–T119E–DP

10 9 8 7 6 5 4 3 2

Printed in the United States of America

CONTENTS

PREFACE xi

1 INTRODUCTION: PROFESSIONAL SYSTEM MANAGEMENT 1

1.1 Chapter Objectives 1
1.2 Introduction 1
1.3 Objective-based Management 2
1.4 Communication Skills 4
1.5 Logical Troubleshooting Skills 5
1.6 Conclusion 5

2 SITE CONFIGURATION 7

2.1 Chapter Objectives 7
2.2 Introduction 7
2.3 Site Attributes 8
2.4 Disaster Planning 14
2.5 Summary 15
2.6 Additional References 15

3 SYSTEM CONFIGURATION 16

3.1 Chapter Objectives 16
3.2 OpenVMS System Disk Organization 16
3.3 System Logical Names 18
3.4 VMScluster Configurations 18
3.5 Additional References 24

4 BOOTING OpenVMS 25

4.1 Chapter Objectives 25
4.2 Introduction 25
4.3 Console Subsystems 26
4.4 Configuring a Boot Device 30
4.5 Boot Methods 35
4.6 SYSBOOT Options 39
4.7 Interrupt Priority C 42

4.8 Building Stand-alone Backup Kits 43
4.9 Creating Alternative System Disks 43
4.10 Conclusion 44
4.11 Additional References 44

5 SYSTEM STARTUP PROCEDURES 45
5.1 Chapter Objectives 45
5.2 Introduction 45
5.3 System Startup Sequence 46
5.4 Site-specific Startup Procedures 50
5.5 Coordinating Site-specific Startup Procedures 53
5.6 Coordinating Shareable System Files 58
5.7 Miscellaneous Startup Issues 60
5.8 Optimizing System Startup 61
5.9 Conclusion 66
5.10 Additional References 66

6 SYSTEM SHUTDOWN 68
6.1 Chapter Objectives 68
6.2 Introduction 68
6.3 Orderly Shutdown Issues 69
6.4 Orderly Shutdown Procedures 76
6.5 Emergency Shutdown Procedures 82
6.6 Automating Shutdown 86
6.7 Conclusion 87
6.8 Additional References 88

7 LICENSE MANAGEMENT 89
7.1 Chapter Objectives 89
7.2 Introduction 89
7.3 Digital Software Licenses 91
7.4 License Types 92
7.5 License Management Facility (LMF) 95
7.6 Additional References 102

8 QUEUE MANAGEMENT 103
8.1 Chapter Objectives 103
8.2 Introduction 103

8.3 Queue Types 104
8.4 Queue Manager 106
8.5 Controlling the Queue Manager 108
8.6 Queue Naming Conventions 111
8.7 Configuring Printer Devices 112
8.8 Creating and Starting Queues 117
8.9 Other Queue Characteristics 121
8.10 Device Control Libraries 121
8.11 Print Forms 124
8.12 Additional References 126

9 MANAGING USER ACCOUNTS 127
9.1 Chapter Objectives 127
9.2 Introduction 127
9.3 User Authorization Database 127
9.4 Account Characteristics 129
9.5 Creating Template Accounts 138
9.6 Managing Accounts 138
9.7 Tools 140
9.8 Additional References 141

10 DISK SPACE MANAGEMENT 142
10.1 Chapter Objectives 142
10.2 Introduction 142
10.3 Backup Strategy 142
10.4 Disk Cluster Size 143
10.5 OpenVMS Purge Utility 143
10.6 File Version Limits 145
10.7 Disk Quotas 148
10.8 Using File Expiration Dates 152
10.9 Volume Rebuild 156
10.10 Archiving System Log Files 157
10.11 Tools 158
10.12 Additional References 158

11 LOGIN COMMAND PROCEDURES 159
11.1 Chapter Objectives 159
11.2 Introduction 159

11.3 Login Sequence 160
11.4 System Login Command Procedure 160
11.5 User Login Command Procedure 168
11.6 Reducing Login Time 169
11.7 Establishing Menus 173
11.8 Captive Login Procedures 174
11.9 Additional References 175

12 AUTOMATING SYSTEM MANAGEMENT 176
12.1 Chapter Objectives 176
12.2 Introduction 176
12.3 Error Handling 176
12.4 Restart Handling 179
12.5 Synchronizing Batch Jobs 181
12.6 Scheduling Batch Jobs 182
12.7 Tools 184
12.8 Additional References 185

13 PERFORMING BACKUPS 186
13.1 Chapter Objectives 186
13.2 Introduction 186
13.3 Backup Requirements 187
13.4 Backup Techniques 190
13.5 Developing a Backup Strategy 194
13.6 Backup Tools 194
13.7 Implementation 199
13.8 OpenVMS Backup Performance 206
13.9 Tools 207
13.10 Additional References 207

14 GETTING HELP 208
14.1 Chapter Objectives 208
14.2 Introduction 208
14.3 Developing a Site Guide 208
14.4 Digital Documentation 210
14.5 Digital Support 216
14.6 DECUS 221
14.7 Conclusion 223

15 SYSTEM INTEGRITY MONITORING 225
15.1 Chapter Objectives 225
15.2 Introduction 225
15.3 Monitoring Hardware Errors 225
15.4 Monitoring Disks 228
15.5 Monitoring Processes 229
15.6 OPCOM 233
15.7 Tools 234
15.8 Additional References 234

16 MAIL MANAGEMENT 235
16.1 Objectives 235
16.2 Introduction 235
16.3 System Profile Database 235
16.4 User Mail Database 237
16.5 Managing Mail Profiles 238
16.6 Mail Logical Names 239
16.7 Forcing Message Count Synchronization 240
16.8 Managing Mail Distribution Lists 240
16.9 Using MAIL as a Management Tool 241
16.10 Tools 242
16.11 Additional References 243

17 SECURITY MANAGEMENT 244
17.1 Chapter Objectives 244
17.2 Introduction 244
17.3 Controlling System Access 245
17.4 Object Protection 254
17.5 Security Class Object Protection 271
17.6 Protected Subsystems 272
17.7 Security-related System Parameters 275
17.8 Network Security 277
17.9 OpenVMS Security Auditing 281
17.10 Additional References 285

18 PERFORMANCE MANAGEMENT 286
18.1 Chapter Objectives 286
18.2 Introduction 286

18.3 Identifying Performance Objectives 287
18.4 Monitoring Application Performance 289
18.5 Identifying Resource Dependencies 295
18.6 Load Balancing 302
18.7 Tuning Memory 305
18.8 Tuning I/O 316
18.9 Tuning CPU 321
18.10 Expanding Capacity 324
18.11 Sample Performance Problem 324
18.12 Managing System Parameters 325
18.13 Additional References 332

19 NETWORK MANAGEMENT 333
19.1 Ethernet Basics 333
19.2 DECnet Introduction 340
19.3 DECnet Components 340
19.4 DECnet Management 344
19.5 Managing LAT (Local Area Transport) 351
19.6 Additional References 366

APPENDIXES

A CONTACT INFORMATION 367

B SITE GUIDE OUTLINE 369

C SYSTEM CONSOLE SUMMARY 372
C.1 QBUS-based Systems 373
C.2 SCSI-based Systems 377
C.3 DSSI-based Systems 379
C.4 XMI-based Systems 381
C.5 ALPHA and VAX 7000/10000 Systems 383
C.6 UNIBUS-based Systems 385
C.7 BI-based Systems 388

D OpenVMS PRIVILEGES 390

E DISKETTE CONTENTS 394

INDEX 419

PREFACE

This book describes system management procedures for Digital systems running the OpenVMS operating system. Unless otherwise noted, discussions are applicable to OpenVMS VAX Version 5.5. However, within each chapter, new features that were introduced with OpenVMS VAX Version 6.x are noted. In addition, most discussions that are applicable to OpenVMS Version 5.5 are also applicable to Digital Alpha systems running OpenVMS AXP Version 1.5. As of version 6.1, the functionality of OpenVMS on Alpha and VAX is nearly identical.

Existing manuals and publications serve as reference tools for OpenVMS commands and utilities; they describe what each OpenVMS facility *can* do. In contrast, this book is an examination of system management philosophies, techniques, strategies, and tools; it describes not only what *can* be done but also what *should* be done, *why*, and *how*. Wherever possible, example Digital Command Language (DCL) procedures are provided to facilitate implementing the strategies discussed. These sample procedures are contained on the diskette included with this book.

Background

Computer systems have come a long way in terms of reliability and maintainability; however, they do not run themselves. They require a staff of skilled systems personnel to handle day-to-day problems and to ensure the long-term delivery of quality service. This is the job of the system manager. Without a system manager, machine resources are eventually exhausted, system software and applications become outdated, reliability begins to deteriorate, performance degrades, and eventually the system ceases to function. Comprehensive system management is essential.

Unique OpenVMS Management Problems

The nature of Digital OpenVMS systems presents some unique system management challenges. Because they are mini-class computer systems, OpenVMS systems are frequently used in small and medium-sized companies

or within departments of larger organizations. Historically, OpenVMS systems support fewer applications and users than do large, corporate mainframe data centers. Even though OpenVMS installations operate on a smaller scale, the management requirements of any OpenVMS site are no different from those of the largest mainframe data center. Both require highly skilled personnel to perform backups, performance tuning, user account maintenance, security auditing, preventive maintenance, capacity planning, and the hundreds of other system management tasks.

This misconception that OpenVMS systems "run themselves" commonly results in the formation of a system management staff as an afterthought. Frequently, engineers, programmers, secretaries, and office managers are one day told, "Guess what? You're the new OpenVMS System Manager!" Of course, they retain all of their previous responsibilities as well.

In larger organizations the economies of scale allow system management tasks to be delegated among several specialists, such as a system programmer, security manager, or performance specialist. In many OpenVMS environments you are expected to perform all these tasks yourself. Or even worse, you may be expected to perform these tasks on a part-time basis.

Doing More with Less

In addition to the underestimation of OpenVMS system management complexities, the "down-sizing" trend of the 1990s has increased pressure to reduce data processing costs. This requires doing more with smaller budgets and fewer people. Some have approached this problem by planning to create "lights-out" or totally automated, unstaffed data centers. Although this concept is exciting, it should be the final stage of site automation, not the first.

When personal computers were first introduced, businesses enthusiastically embraced them as a means to automate their accounting and bookkeeping functions. However, computers did absolutely nothing for companies that had less-than-organized bookkeeping strategies and procedures to begin with. The point is that "lights-out" technologies can only be considered *after* sound, efficient, and proactive management strategies have been carefully thought out and implemented. Although this book is not a guide to automating Digital installations, it discusses the fundamental strategies that are a prerequisite to data center automation.

Purpose

I learned OpenVMS system management by spending thousands of hours reading OpenVMS documentation, attending conferences and local user group meetings, talking with friends, and constantly logging technical support calls. For many people, learning OpenVMS is a similar, arduous experience. This has been the only way to learn OpenVMS system management because no formal training is available. There are no college courses entitled System Management 101. There are plenty of week-long training seminars, but they are designed to present a maximum amount of information in a minimum amount of time. You simply cannot learn system management by spending a couple of days in a training class.

As in many disciplines, the best system management training occurs in the field. However, when you're in the field, it helps to have a guide. OpenVMS documentation is voluminous, but when specific system management issues are discussed, they are examined only in general, and examples are too simple. With more than 20 volumes, and thousands of pages, it's easy to get lost. This book attempts to be a sorely needed guide and is a collection of the philosophies, techniques, strategies, and tools I've learned and used over the years. These things were not learned by reading manuals or going to seminars—they were learned in the trenches of system management.

I wrote this book so that others could broaden their system management skills with the help of my experiences and, I hope, with a little less pain. I also hope that this book will help clarify what an enormous and important job system management is so that it will be given the attention and respect it deserves. In many environments, systems can be *maintained* with a minimal staff, but a serious commitment of resources is necessary to properly *manage* systems and meet critical business needs.

Audience

This book is directed at people who, regardless of their "real" jobs, are responsible for day-to-day management of OpenVMS systems. This is *not* an introductory text—a basic understanding of OpenVMS, DCL, and system management concepts is assumed. This book is most appropriate for people with at least 6 months of system management experience or advanced system management training. However, even the most experienced reader will find information, strategies, and tools that are extremely valuable.

Organization

The introduction establishes the *purpose* or *philosophy* of system management. The central theme of this philosophy is to clearly define *service-level objectives* for your site that define the nature of services you must provide to meet the objectives of your organization.

The remainder of this book is divided into chapters that focus on individual system management tasks. At the beginning of each chapter, a set of objectives outlines what the chapter hopes to accomplish. The content of each chapter discusses how the task presented is related to meeting the service objectives, strategies for meeting those objectives, and tools that facilitate implementation. The last section of each chapter provides additional references about the subject.

Chapters are ordered chronologically in terms of the life of an OpenVMS system. Chapters 2 through 8 present tasks associated with setting up and configuring a system. You may have already completed many of these tasks, but you should review these chapters to ensure that your configuration is compatible with your service-level objectives. Chapters 9 through 19 discuss tasks that are necessary to keep the system operating once it has been configured.

In order to make OpenVMS commands and responses easier to identify, I have presented them using a monospace font. For example:

```
$ DIR LETTER.DOC/DATE=MODIFIED/SIZE
```

Boldface text is used to represent user input when additional emphasis is required. For example:

```
How many minutes until final shutdown [0]: 10
```

Suggestions

You should begin by reading the Introduction and thinking in terms of service-level objectives. Keep in mind the objectives of your organization and then focus on the chapters you consider most relevant to meeting those objectives.

1 INTRODUCTION: PROFESSIONAL SYSTEM MANAGEMENT

1.1 Chapter Objectives

- Apply classic business management strategies to system management
- Define *service-level objectives* and their importance
- Focus on overall approach to effective system management

1.2 Introduction

The term *system manager* is ambiguous. What exactly does a system manager do? What or whom does a system manager manage? Is the position technical or managerial? Frequently, job descriptions lack or avoid these details and leave a default expectation—do "whatever is necessary" to keep systems running.

To accomplish this, system managers generally perform tasks such as configuring new hardware, performing software upgrades, managing system startup procedures, tracking software licenses, configuring print and batch queues, managing user accounts, monitoring disk space, ensuring regular system backups, monitoring system security, and maintaining system performance. Although these tasks require technical skill, the process of applying them to the objectives of an organization is more of a managerial task. Thus, the optimal system manager is a composite technician and manager—a *techno-manager*. To achieve this standing requires the following nontechnical skills:

- Objective-based management

- Good communication skills
- Logical troubleshooting skills

1.3 Objective-Based Management

Users complain that systems run slowly, or are down frequently. Management insists that you are spending too much money. This leaves you caught in the middle—you have to placate frustrated users with contracting resources.

Peter Drucker, the classic management scientist, frequently wrote about the idea of *management by objective*. This strategy is based on the idea that you can only operate effectively when you understand business objectives and direct your efforts toward meeting them. This concept applies well to the process of system management. The enormity of the task leaves many system managers unsuccessfully scrambling to meet undefined or unattainable user and management expectations. Clear objectives developed in collaboration with users and management will help you better focus your system management efforts, moderate user expectations, and justify costs.

The first step is to clearly define exactly what the objectives are. Almost everyone agrees that system managers are supposed to keep systems running. But what does this mean? Exactly when should systems be up? How fast should they perform? At what price? You may find it helpful to start by defining an overall *mission statement* that sums up what everyone agrees should be accomplished. For example, this book focuses on the following mission statement:

> *To keep applications available at a level of performance that meets user expectations*

After you've developed a mission statement, develop more detailed *service-level objectives* that support the mission, for example:

System uptime

Target *scheduled* downtime (in hours/year) (for upgrades, preventive maintenance)

Target *unscheduled* downtime (in hours/year)

User requests (time to service)

File restore from backups

Add user account

Answer general technical questions

Application requests (time to service)

Mount tapes

Change printer forms

Data integrity

Maximum data loss due to system failure (in minutes, hours, or days)

Software upgrades

How frequently should products be upgraded? (OpenVMS, layered products, etc.)

Performance

For each key application specify the target response time

Security

Agree on level of protection against:

User error
Internal attacks
External attacks

To be valuable, you must be able to measure your ability to achieve each service objective. Whenever possible, quantify exactly what the goal is. For example, you can calculate system uptime by using the following formula:

Uptime % = [(Hours per year – downtime) / Hours per year] * 100%

If a system were down for 50 hours during a year, uptime would be as follows:

Uptime % = [(365 * 24 – 50) / 365 * 24] * 100% = 99.43%

On a regular basis you should post both service objectives and statistics indicating the level of service you actually provided.

1.4 Communication Skills

Although system management is technically oriented, excellent personal skills and a positive attitude are essential. An OpenVMS guru with poor communication skills will never be a good system manager. On the other hand, a novice OpenVMS technician with excellent communication skills, patience, and an ability and desire to learn can become a great system manager with time.

System management requires frequent, sometimes stressful communication with many groups within an organization. To be effective, you must be even-tempered, well-organized, and extremely patient. Your tone must be noncondescending—you need to be able to educate people without making them feel dumb.

Communication is a two-party event. To communicate effectively you need to know your audience. You need to communicate at a level appropriate to the expertise of your audience. Don't try to impress people with your knowledge, yet don't insult the more experienced user. *Don't try to explain what's wrong to someone who only cares when it will be fixed.* Understand your user's objectives and provide only relevant information.

Sometimes you tend to feel users exist only to make your life miserable. Actually, users are your reason for existence. If it weren't for users, you would not have a job. If you don't like dealing with users, you are in the wrong profession.

Your door should always be open (if you're lucky enough to have a door). Sometimes it's very hard to concentrate when people are constantly calling and asking questions. However, the time you spend helping them now is far less than the time required to clean up the results of unanswered questions. Nigel Sharp illustrated this point:

> . . . it should be clear that you can be bothered, and that in many cases you would prefer to be bothered. A nervous student once ruined one of our printers because he didn't want to disturb anyone. (He simply added toner, since that's what it said it needed.) Unfortunately, he poured liquid toner into a dry toner container . . . [1]

Users sometimes understand that it will take time for you to solve their problem. However, they should never have to wait to tell you about a problem. Whether you have a secretary, Voice Mail, answering machine, or electronic mail, there should always be a mechanism that allows users to imme-

diately log problems. Imagine how you would feel if you had to call a service department ten times before someone even answered.

If you already think you're good, you'll never get better. If there were people who knew everything there was to know about Digital software and systems then the Digital software support center would be staffed by a few dozen people. There are no such experts, and it takes hundreds of people to provide a pool of knowledge sufficient to support OpenVMS. The true experts are modest about what they do know. Most important, they know how to learn what they don't know.

1.5 Logical Troubleshooting Skills

You must be a logical thinker and an expert troubleshooter. You have to be able to transcend a typical user response (e.g., "it's broken") by asking questions that will identify precisely what "it" is. Avoid asking users questions that are too easily answered without much thought. For example, don't ask, "Has this always worked?" Everyone will usually say "Yes." Rather, ask, "*When* was the last time this worked (or didn't work)?"

In an emergency, you may have to temporarily *fix* a problem. However, you should always go back later and permanently *solve* the problem. Solving a problem means understanding why the problem occurred and taking the necessary steps to prevent it from occurring again. Fixing a problem is like smothering a fire by throwing a huge pile of leaves on it—the flames temporarily disappear but soon burst into an inferno twice as large as before. Solve problems—don't go searching for even larger piles of leaves.

Some problems cannot be immediately solved. For example, if disk drives run out of space on a weekly basis, the solution might be to purchase more disk space, but that could take several months. In the meantime, you must prevent service from being denied to users. At a minimum, you should deploy a disk space monitoring utility that proactively notifies you when disk space is low so you can purge files before users are affected.

1.6 Conclusion

If you are to have a clear focus, you must demand objectives or develop them yourself. Operating without objectives is attempting to achieve the impossible—infinite service-level expectations. Once you have set objectives, system management is nothing more than classic customer service. As with any job

that involves direct customer (user) contact, excellent personal and communication skills are essential. If you can work within these guidelines, you'll be well on your way to becoming an effective system manager—a professional system manager.

Notes

1. Nigel Sharp, *The Philosophy of System Management*, DECUS Transactions (Fall 1991), pp. 1–4.

2 SITE CONFIGURATION

2.1 Chapter Objectives

- Determine site configuration required to meet service-level objectives
- Explore physical, electrical, environmental, and other factors that characterize site configuration
- Suggest management policies and operational procedures necessary to maintain functioning site
- Briefly discuss disaster planning issues

2.2 Introduction

Within the context of this chapter, *site configuration* refers to the physical, environmental, electrical, safety, and operational characteristics of the space immediately surrounding your computer systems.

- *Physical* characterizes the size, shape, and location of the room or rooms that contain your systems (e.g., the computer room)
- *Environmental* describes computer room attributes, such as temperature, humidity, air flow, and existence of airborne contaminants
- *Electrical* addresses the quality and availability of electrical power from the local utility company as well as in-house equipment used to enhance power
- *Safety* focuses on the equipment necessary to ensure safety of personnel, equipment, and building structures
- *Operational* discusses the procedures and policies used to manage, maintain, and operate the site components

These site configuration features represent the foundation that supports the systems and the applications built upon that foundation. Computer

equipment is generally more reliable than mechanical equipment but only when precise temperature, humidity, air flow, air quality, and power-level tolerances are maintained. When site support systems fail, any system contained within the site is vulnerable to failure, or at best will operate less reliably.

Not many system managers have an opportunity to be involved in the initial design and construction of the data center they manage. More than likely, you inherited a site that was configured and constructed by your predecessors. For this reason, you might take for granted the configuration of your site and assume it is acceptable. Or, you might be tempted to ignore the issue altogether and consider it someone else's responsibility.

Site configuration represents a foundation pivotal to achieving service-level objectives. You cannot afford to make any assumptions, and you certainly cannot ignore it. You must ensure that current site characteristics match the needs of the systems they support. You also need to consider whether your site configuration needs match changing system configuration or service objectives. For example, as you add computer hardware, you may need to add cooling and power conditioning. Similarly, a change in business objectives might require redundant air conditioners or higher capacity power backup.

2.3 Site Attributes

The following sections discuss the five aspects of site configuration in more detail.

2.3.1 Physical

Location

Computer rooms are best located in the center portion of a building, away from exterior windows. This provides additional security and protects the equipment from the effects of exterior temperature changes. Systems should be located away from vibration (e.g., heavy industry, airports), sources of electrical interference (e.g., welders, large motors), and overhead flood sources (e.g., toilets, water-chilled air conditioners, sprinkler systems).

Structural Capacity

Building floors must have the structural capacity to support the weight of equipment, supplies, and personnel.

Flooring

A raised floor is essential for many installations. Many Digital systems and disk subsystems require specific air flow rates for cooling purposes. Raised floors meet these requirements by routing air below the floor and up through vents beneath or immediately behind equipment. If you have more than a few systems or peripherals, a raised floor allows you to more easily isolate and route data and power cables.

The height of the raised floor must be adequate to house the cables that run beneath it and still have additional space for good air flow. Floor tiles themselves should be antistatic and provide a firm footing to prevent slips and falls. Falling on a disk drive is painful to both humans and disk heads alike.

Equipment Layout

The layout of equipment should facilitate access by service personnel, minimize cable lengths and routing difficulties, maximize air flow, and allow for future reconfiguration. Printers should be located in a separate room to minimize dust generation.

2.3.2 Electrical

Power Basics

More than likely, your electric power is delivered to your site from your local utility company. Power from generating stations flows into a series of grids that satisfy the needs of buildings attached to the grid. As demand changes on these grids, the utility company shifts power to meet those needs. Some locations (usually business areas) are fed by multiple grids. If one grid fails (e.g., a car runs into a utility pole) the remaining grids can pick up the load. These load-switching events can result in surges or drops in power, which some equipment cannot tolerate.

There are two basic concerns with electrical power: *availability* and *quality*.

Availability is the percentage of time power is available to equipment. Sometimes events such as fire, flood, wind, or lightning induce failure of all possible power paths to a location, causing an outage. Other times, load-switching causes a momentary loss of power. Either of these events may cause unprotected computer systems to crash and require a complete reboot.

Some Digital systems can be optionally outfitted with a main memory battery backup. The battery backup preserves the contents of main memory for a short period of time so that if power is restored quickly the system can

sometimes fully recover. Unfortunately, battery backup provides power only to main system memory—other peripherals such as disks and terminals lose power, rendering the system inoperable during the outage. In addition, devices such as disks do not react well to sudden power loss and may become damaged as a result of the outage. Therefore, even if the main central processing unit (CPU) recovers from the outage because of the memory battery backup, other equipment may be damaged, causing downtime anyway. Also, if data transactions are interrupted by a power failure, data may become corrupted—again requiring system downtime.

Power availability can be enhanced by using an uninterruptible power supply (UPS). A UPS consists of a series of high capacity batteries that provide backup power if the local utility fails. As long as all system peripherals are attached through the UPS, the system can continue to operate for as long as battery capacity allows. You cannot protect all remote terminals with a UPS, thus users may not be able to access the system during an outage. However, a UPS gives you the time to ensure that key transactions are completed and allows an orderly system shutdown. This minimizes the risks of outage-induced hardware failures and data corruption.

If your site requires protection against extended power failures, some type of backup generator system may be necessary. With a generator, power can be maintained for as long as available fuel allows.

Power quality measures the consistency of electrical power in terms of voltage fluctuations and power-line noise. Most Digital systems can tolerate voltage fluctuations of ±10% but can tolerate only a minimal amount of noise. Power-line noise is usually caused by the arcing of electric current in relays, motors, lights, and welders. Computer equipment can be protected from such noise with an *isolation transformer* between computer loads and possible noise-generating loads. Voltage fluctuations can also be caused by utility load-switching as well as localized load-switching caused by heavy equipment. You can isolate equipment from voltage fluctuations by installing a power conditioning system, which also contains an isolation transformer.

For the most part, local utility companies do an excellent job and have sufficient backup systems to ensure reliable service for most situations. Some critical applications, however, may require a UPS or power conditioning equipment or both to meet service objectives. In some areas where power anomalies are frequent because of local weather patterns, additional power protection may be prudent. If you are unfamiliar with the power characteristics in your area, contact your local utility and ask for a history of voltage fluctuations and outages.

Electromagnetic Interference

Another area of concern is electromagnetic interference (EMI). When current flows through a wire, electromagnetic fields develop around the wire. The stronger the current, the larger and stronger are these fields. When equipment or data cables are subjected to magnetic fields, electronic signals can be disrupted. Other sources of electrical disturbances, such as static or lightning, can cause equipment damage.

Current-induced EMI problems can be minimized by using cabling with sufficient shielding and grounding. In addition, data cables should be routed separately from EMI-generating power cables. Where power cables cross data cables, they should be oriented at right angles to minimize the effects of EMI.

Electrostatic Discharge

Protection from static discharge can be enhanced by minimizing the use of carpets and by maintaining proper humidity levels. If static is a problem, static discharge mats can be installed at the entrances to the computer room. Lightning is simply a severe form of electrostatic discharge. Although you cannot completely eliminate the risk of equipment damage from lightning, you can greatly enhance protection with proper building grounding and lightning arrestors. In geographic areas where lightning is common, you should carefully assess your protection against such disturbances.

2.3.3 Environmental

Temperature

The published operating temperature range for most Digital equipment is 70°F ± 5°F).[1] Absolute temperature is important, but the rate of temperature change must also be minimized to 5.4°F/hour.[1] Most important, these temperature ranges must be maintained at the inlet to equipment cabinets, *not* on the wall away from equipment. Air vents must be configured to provide sufficient air flow through or in the immediate vicinity of equipment cabinets.

Humidity

Most Digital equipment is designed to operate at 50% relative humidity ±10% with a maximum change rate of 6%/hour.[2]

Corrosion and Contamination

In some process manufacturing environments, air can become contaminated with chlorine, sulfur, oxides, acids, or dust, all of which can cause premature

failure of electronic equipment. Make sure that computer room air is sufficiently filtered.

2.3.4 Safety Equipment

Power

The main safety device for electric power is an emergency power off (EPO) switch, which should be located at the main exits of the computer room. This provides a mechanism to quickly remove power from the entire computer room in the event of a serious fire, flood, or electrical problem.

Fire

The first level of fire protection is *prevention*. Fires need fuel to burn, so everything you can do to minimize the amount of combustible materials around computer equipment minimizes the possibility of a fire. Walls, floors, ceilings, doors, ceiling tiles, and furniture should be noncombustible or fire-resistant. Use of paper should be kept to a minimum, and paper should be disposed of in noncombustible trash containers. Any highly combustible materials (such as tape head cleaner) should be stored away from the computer room.

The second level of fire protection is *detection*. Smoke detectors should be deployed not only in the computer room but also in surrounding rooms. Within the computer room, detectors should be placed in all major air spaces, including the subfloor, ceiling, and the computer room itself. Alarms should sound within the computer room, a security office, and the local fire department.

The third level of fire protection is *suppression*. At a minimum, hand-held extinguishers should be located at all exits and strategically throughout the computer room. Other fire suppression devices such as Halon, carbon dioxide, and water sprinklers may be necessary for increased fire protection. You should examine the safety, equipment damage, and environmental factors associated with each of these products.

Water

Water detectors should be placed at the lowest point of the subfloors, where water is most likely to collect. Alarms should be located in the computer room and in security offices.

Security

Once someone has physical access to your computer room, he or she can disrupt power, destroy equipment, and remove confidential information. In other words, this person can bypass any software security you might have in place. If your information is valuable or confidential, you need to consider restricting physical access.

Access to the computer room can be restricted by fitting doors with keyed locks, cipher locks, or card readers. Keyed locks are the least expensive, but it is difficult to track who has keys. Cipher locks are activated by entering a combination on a series of buttons. This eliminates the need for keys but requires that combinations be changed whenever key personnel leave your organization. Card readers accept cards with a magnetic strip indicating who is attempting access. Readers can be placed at each entrance to the computer room. A central computer database tracks who is allowed access to which rooms and at what times. This system can be especially helpful to determine who entered the computer room and when; however magnetic cards can be duplicated.

2.3.5 Operational

Procedures

Safety and protection equipment is only as useful as the procedures that describe when and how to operate it. Comprehensive training should be given to all personnel who will be required to interpret equipment alarms, use fire suppression equipment, or activate EPO switches. Specific procedures should be in place that explain how to respond to each type of situation. For example, you probably don't want the EPO switch pressed in response to a fire drill. In some cases, protection equipment causes more system downtime than it saves, because of inadequate operating procedures.

Cleaning Procedures

Dust can reduce the air flow to computer systems and it can build up on system components. Both result in increased component temperature and failure rates. In addition, excessive dust can trigger certain types of smoke alarms. Computer room floors and subfloors should be cleaned with special vacuums that reduce dust exhaust. Air-conditioning and computer air filters should be checked and replaced on a regular basis. Floors can be damp

mopped with water only, and antistatic waxes can be applied if necessary. Conventional waxes can lead to static buildup, and other chemicals can cause air contamination.

2.4 Disaster Planning

No matter how redundant your site support systems are, events such as earthquakes, fires, floods, hurricanes, or terrorist acts can occur that in turn cause site failure. Any event that causes extended site failure can be considered a disaster.

As discussed in chapter 1, the service objectives at your site should specify the nature of service that you need to provide in the event of a disaster and how quickly you need to restore service. In general, you should have an alternative site and equipment that are geographically isolated from any possible localized disaster. Depending on these requirements, you may need to implement one of the following site backup strategies.

2.4.1 Backup Site

A backup site generally consists of duplicate computer equipment at a second geographic location. In the event of a primary site failure, you can retrieve your backups from off-site storage and restore them at the backup site. Depending on the quantity of data to be restored, you can normally restore service within a day. However, you would only be able to provide a system that was as current as your latest backups.

2.4.2 Hot Site

A hot site is similar to a backup site, except that backups are continually restored to the backup site so that contingent service can be made available much more quickly. With a hot site, some level of service can usually be restored in a matter of hours.

2.4.3 Distributed Site

The highest level of disaster tolerance can be achieved by establishing a *distributed site*. A distributed site consists of two physically distant redundant data centers linked by a network. All system transactions are automatically mirrored at both sites instantaneously. In the event that one site fails, the other site is immediately available to provide disaster backup. This strategy

allows service to be restored in a matter of seconds or minutes, with the most recent data available. This type of configuration is possible utilizing the Digital Multi-Datacenter Facility (MDF) product within a VMScluster environment.

2.5 Summary

The first site I ever managed consisted of two VAX 11/785 systems crammed into a narrow computer room that did not have a raised floor. The equipment was no more than three feet away from the walls, so Digital Field Service could hardly open the cabinet doors, never mind get behind the machines. Cables were routed in rain gutters from the local hardware store. Power, network, terminal, and data cables commingled extensively. The equipment was cooled by two water-chilled air conditioners, the water supply running above the ceiling tiles. Whenever water was shut off in the building, a pressure valve in one of the air conditioners would shut off the water supply until it was manually reset. Because the capacity of the remaining air-conditioning unit was insufficient to cool the entire room, temperatures would soar until someone noticed how hot the equipment was (usually many hours later). Because there was no power conditioning or UPS, power failures would frequently cause system crashes.

As you can imagine, system reliability was less than optimal. Although I inherited this site, and improved it somewhat, it did not even occur to me at that time that many of the reliability problems we experienced might be due to inadequacies in the site design.

2.6 Additional References

The bulk of this chapter is a summary of the document listed below and those in the Notes section. They are available from Digital.

Building Dependable Systems, chapter 5, "Maintaining a Dependable Environment," Digital, November 1991.

Notes

1. Digital Equipment Corporation, *Site Environmental Preparation Guide,* November 1990 (Order number #EK-CSEPG-MA-002), p. 63.
2. Digital Equipment Corporation, *Site Environmental Preparation Guide,* November 1990, p. 63.

3 SYSTEM CONFIGURATION

3.1 Chapter Objectives

- Examine the organization of an OpenVMS system disk
- Review standard OpenVMS logical names
- Present basic VMScluster configuration options

3.2 OpenVMS System Disk Organization

By default when you install OpenVMS, a directory structure is created on the target device, as follows:

```
[000000]                    ! Master File Directory

    [SYS0.]                 ! SYS0 system-specific root
        [SYSEXE.]           ! Specific system files
        [SYSMGR.]           ! Specific management files
        ...
        [SYSUPD.]
        [SYSCOMMON.]        ! Link to common root (VMS$COMMON)

    [VMS$COMMON.]           ! Cluster common directory
        [SYSEXE.]           ! Common system files
        [SYSMGR.]           ! Common management files
        ...
        [SYSUPD.]
```

If the system disk is used as a VMScluster common system disk, an additional system-specific root is created for each cluster member you add (e.g., with CLUSTER_CONFIG):

```
    [SYS1.]                 ! First system-specific root
        [SYSEXE.]
```

```
        [SYSMGR.]
        ...
        [SYSUPD.]
        [SYSCOMMON.]

    [SYS2.]                    ! Second system-specific root
        [SYSEXE.]
        [SYSMGR.]
        ...
        [SYSUPD.]
        [SYSCOMMON.]
```

The directories under each system-specific root contain only the files that cannot be shared in a cluster (e.g., system parameter files, operator log files). All other OpenVMS files are located under the cluster-common root VMS$COMMON. For example, all executable images (e.g., DIRECTORY.EXE) are located in [VMS$COMMON.SYSEXE]. Because of restrictions in bootstrap code, an OpenVMS system must boot by referring to the system-specific root (e.g., [SYS0]). This presents a problem because the files needed to boot are actually stored in the root directory [VMS$COMMON]. To handle this problem, there is a directory entry in each system-specific root (SYSCOMMON.DIR) that is an alias for the common root ([VMS$COMMON]). This allows OpenVMS to locate boot files in [VMS$COMMON] by referring to [SYSx.SYSCOMMON].

Using the DIR/FILE command you can confirm that the file identification numbers of both of these directory files are the same—they *are* the same file:

```
$ DIR/FILE BIRCH$DUA1:[SYS0]SYSCOMMON.DIR ,-
BIRCH$DUA1:[000000]VMS$COMMON.DIR

Directory BIRCH$DUA1:[SYS0]

SYSCOMMON.DIR;1  (11,1,0)

Total of 1 file.

Directory BIRCH$DUA1:[000000]

VMS$COMMON.DIR;1  (11,1,0)

Total of 1 file.

Grand total of 2 directories, 2 files.
```

3.3 System Logical Names

To facilitate access to system directories (e.g., [SYS0.SYSEXE]), OpenVMS defines a number of system-wide logical names. At the lowest level, the logical name SYS$SYSDEVICE is defined as the device name of the system disk, for example:

```
$ SHOW LOGICAL SYS$SYSDEVICE
SYS$SYSDEVICE = "BIRCH$DUA1:"
```

At the next level, the logical names SYS$SPECIFIC and SYS$COMMON locate the system-specific and the cluster-common root directories, as follows:

```
$ SHOW LOGICAL SYS$SPECIFIC
  "SYS$SPECIFIC" = "BIRCH$DUA1:[SYS0.]"
$ SHOW LOGICAL SYS$COMMON
  "SYS$COMMON" = "BIRCH$DUA1:[SYS0.SYSCOMMON.]"
```

Next, the logical name SYS$SYSROOT is defined as a search list that points to both the system-specific and cluster-common roots, as follows:

```
$ SHOW LOGICAL SYS$SYSROOT
  "SYS$SYSROOT" = "BIRCH$DUA1:[SYS0.]"        ! System-specific root
      = "SYS$COMMON:"
1 "SYS$COMMON" = "BIRCH$DUA1:[SYS0.SYSCOMMON.]"  ! Cluster-common root
```

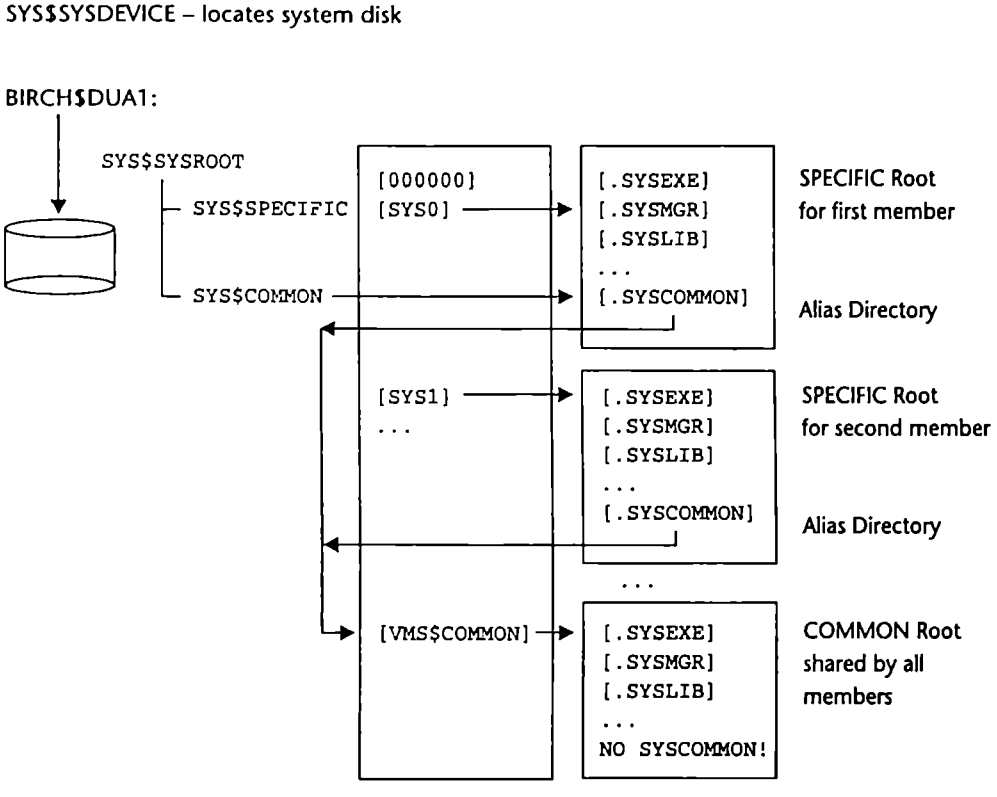
When you access a file using a search list, OpenVMS attempts to locate the file by sequentially checking each directory included in the list. However, when you create a file, OpenVMS creates the file in the first directory listed. Figure 3.1 shows the relationship between the SYS$SYSROOT, SYS$COMMON, SYS$SPECIFIC, and SYS$SYSDEVICE logical names.

OpenVMS also defines logical names for key system directories in terms of SYS$SYSROOT, as indicated in Table 3.1.

3.4 VMScluster Configurations

A VMScluster consists of hardware and software mechanisms that allow several systems to share resources. A cluster generally consists of two or more members that are physically connected by a common bus or by an Ethernet network. VMScluster software allows each cluster member to share disks, tapes, queues, and files. Clusters are categorized based on how tightly they

Figure 3.1 OpenVMS System Disk Logical Structure



are integrated. In a *common-environment* cluster, all members generally boot from a common system disk and present the same applications and devices to users. To a user, a common-environment cluster appears as a single, large system. In a *multiple-environment* cluster, members may or may not boot from a common system disk, but system files are not usually shared, and different applications are started on each system.

The following sections discuss some of the benefits of establishing a VMScluster.

Common System Disk

The most powerful aspect of VMSclusters is the ability to designate a single cluster-accessible disk as a common OpenVMS system disk. All cluster members can then boot from the common disk, allowing them to share one copy of key system files. This facilitates system management because you manage

Table 3.1 OpenVMS System Logical Names

Logical Name	*Equivalence*	*Contents*
SYS$ERRORLOG	SYS$SYSROOT:[SYSERR]	System Error Log File (ERRLOG.SYS)
SYS$EXAMPLES	SYS$SYSROOT:[SYSHLP.EXAMPLES]	Example command procedures and programs supplied by Digital
SYS$HELP	SYS$SYSROOT:[SYSHLP]	Help library files
SYS$LIBRARY	SYS$SYSROOT:[SYSLIB]	Shareable image files and object libraries
SYS$LOADABLE_IMAGES	SYS$SYSROOT:[SYS$LDR]	OpenVMS executive image files and device drivers
SYS$MAINTENANCE	SYS$SYSROOT:[SYSMAINT]	Reserved for Digital hardware diagnostic programs
SYS$MANAGER	SYS$SYSROOT:[SYSMGR]	Site-specific startup procedures, system management procedures
SYS$MESSAGE	SYS$SYSROOT:[SYSMGR]	System message text files
SYS$SHARE	SYS$SYSROOT:[SYSLIB]	Shareable image files and object libraries
SYS$STARTUP	SYS$SYSROOT:[SYS$STARTUP], SYS$MANAGER	Site-independent startup command procedures
SYS$SYSTEM	SYS$SYSROOT:[SYSEXE]	Executable image files for most OpenVMS commands and utilities
SYS$UPDATE	SYS$SYSROOT:[SYSUPD]	Command procedures to perform system updates and maintain system parameters (e.g., VMSINSTAL, AUTOGEN)
SYS$VMSDOC_TEXT	SYS$SYSROOT:[SYSHLP.VMSDOC]	On-line OpenVMS documentation files

only one copy of system start-up files, user authorization files, and queue database files.

Remote File Access

All Digital Command Language (DCL) file manipulation commands (e.g., COPY, CREATE, DELETE, RENAME) can be used to manipulate files throughout the VMScluster. Because all devices appear as local disks, file specifica-

tions are the same as for local files. File transfer is much faster than DECnet, and in most cases access to remote disks is nearly as fast as access to directly attached disks.

Remote Job Submission

You can directly submit a print or batch job to a queue on any cluster member. The submitted job need only exist somewhere in the cluster. This is much simpler than using SUBMIT/REMOTE, in which the command procedure must already exist on the remote DECnet system.

Increased Availability

When a VMScluster is configured as a common-environment cluster, users can log in to any cluster member and transparently access the same disks, files, queues, and applications. System availability is greatly increased, because users can simply log in to another cluster member if the primary system fails.

3.4.1 VMScluster Interconnects

Cluster members can be physically connected with a number of mechanisms. The following sections briefly present several VMScluster configuration options.

Computer Interconnect

Clusters that use the Computer Interconnect (CI) are connected to a Star Coupler in a radial configuration, as shown in Figure 3.2. A single Star Coupler supports a maximum of 8 CI nodes, but you can create an expanded, dual-coupler configuration that supports a maximum of 32 CI nodes per cluster. Typically, in a CI cluster, disks are connected to dedicated storage controllers (e.g., HSC 70), which handle disk I/O requests generated by cluster members. The CI supports a maximum bandwidth of 70 megabits per second; however, the throughput of a single system depends on its adapter.

Digital Storage System Interconnect (DSSI)

Clusters that use the Digital Storage System Interconnect (DSSI) are connected to a DSSI bus in a lateral configuration, as shown in Figure 3.3. The DSSI bus supports a maximum of eight nodes, of which a maximum of four can be OpenVMS systems. The remaining nodes can be Integrated Storage

Figure 3.2 CI VMScluster Configuration

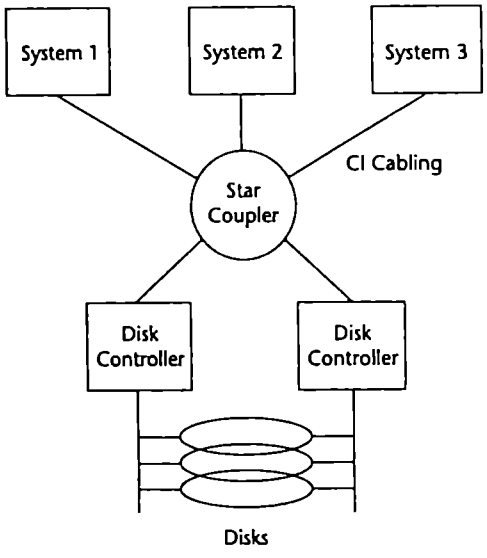
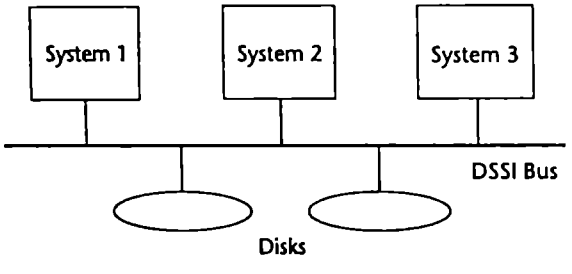


Figure 3.3 DSSI VMScluster Configuration

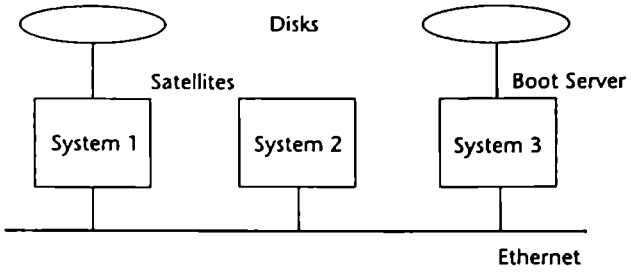


Elements (ISE) disks. The maximum bandwidth of a DSSI bus is 32 megabits per second and is generally limited to a total length of about 25 meters.

Ethernet or Network Interconnect

You can also establish a VMScluster simply by connecting several systems to an Ethernet network, as illustrated in Figure 3.4. Ethernet clusters are also commonly referred to as Local Area VMSclusters (LAVC). In general, one or more satellite members boot from a boot server that contains an OpenVMS system disk. The maximum number of nodes supported in a Network Inter-

Figure 3.4 Network Interconnect VMScluster Configuration



connect (NI) cluster is 96, but this limit is usually further restricted by the meager 10 megabit per second bandwidth of Ethernet.

Fiber Distributed Data Interface

More recently, Digital introduced Fiber Distributed Data Interface (FDDI) as a cluster interconnect, as illustrated in Figure 3.5. FDDI provides considerably more bandwidth than Ethernet (100 megabits per second) and supports a maximum of sixteen FDDI members. In addition, with FDDI, members of the same cluster can be up to 40 kilometers apart.

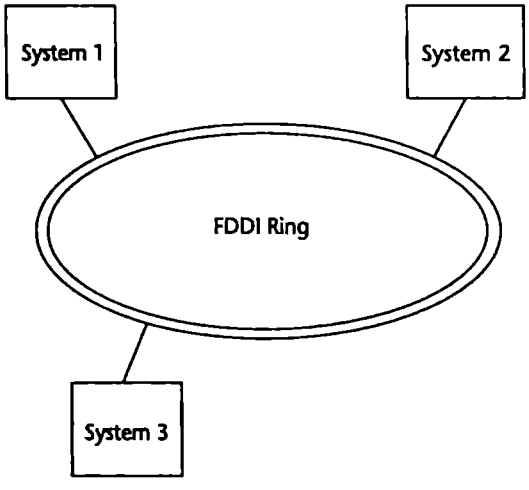
Dual-architecture VMSclusters

A dual-architecture VMScluster is a cluster that contains both OpenVMS VAX and OpenVMS AXP members. It is subject to the following additional configuration restrictions:

- AXP and VAX systems cannot boot from each other
- CI clusters can have either two OpenVMS AXP members or one OpenVMS AXP member and up to three OpenVMS VAX members (VAX 6000, 7000, or 10,000)
- DSSI clusters can have either two OpenVMS AXP members or one OpenVMS AXP member and one OpenVMS VAX member (VAX 4000 Model 100-600)
- All VAX members must be running at least OpenVMS 5.5-2 VAX, and all AXP members must be running OpenVMS 1.5 AXP.

Most of these configuration restrictions have been removed as of OpenVMS version 6.1.

Figure 3.5 FDDI VMScluster Configuration



3.5 Additional References

Davis, Roy G., *VAXcluster Principles*, Digital Press, 1993.
VAX Systems and Options Catalog, DECdirect, 1993.

[OpenVMS 5.0]

Building Dependable Systems, chapter 6, "Dependable Hardware Configurations," Digital, November 1991.
Guide to DECnet-VAX Networking, Digital, April 1988.
VMS Networking Manual, Digital, April 1988.
VMScluster Manual, Digital, November 1991.

[OpenVMS 6.0]

VMScluster System for OpenVMS, Digital, March 1994.
A Comparison of System Management on OpenVMS AXP and OpenVMS VAX, section 2.2.1, Digital, March 1994.

4 BOOTING OpenVMS

4.1 Chapter Objectives

- Provide a single reference source that details boot procedures for most OpenVMS systems
- Discuss procedures for configuring and initiating various boot methods
- Review procedures for configuring console switches and parameters
- Increase system availability by implementing robust boot procedures

4.2 Introduction

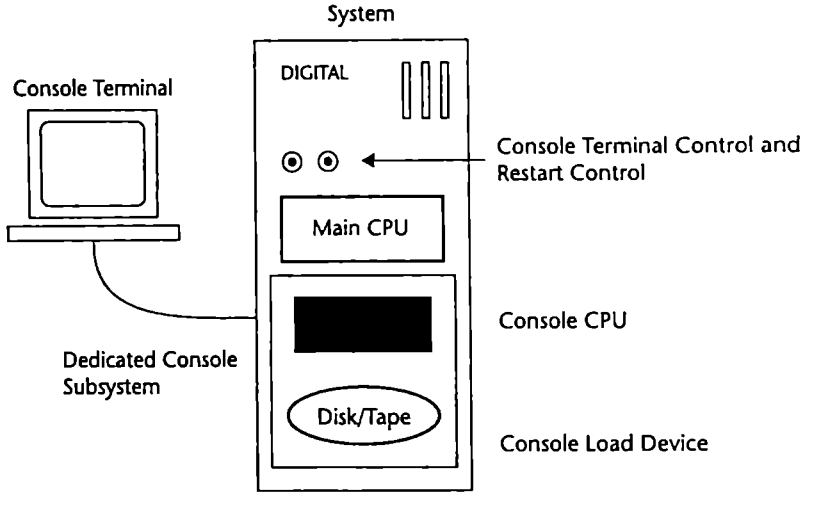
OpenVMS systems normally boot automatically after you power them on or issue the standard BOOT command. As a result, it's easy to take the process of booting for granted. However, things do not always go so smoothly. Sometimes systems hang or will not boot because of misconfigured system parameters, problems with startup procedures, software bugs, or hardware failures. In these situations you may need to use alternative techniques to perform the following procedures:

- Minimal boot
- Conversational boot
- Stand-alone backup boot
- Alternate system disk boot
- Crash dump
- Interrupt Priority C (IPC) utility

Usually you need to employ these techniques when a system is malfunctioning and downtime is accumulating. However, because this doesn't happen very often, even more downtime is incurred while you try to remember

Figure 4.1 Systems with Dedicated Console Subsystems:

VAX11/780, VAX 8600/8650, VAX 88x0, VAX 8500/8700/8800, VAX 9000



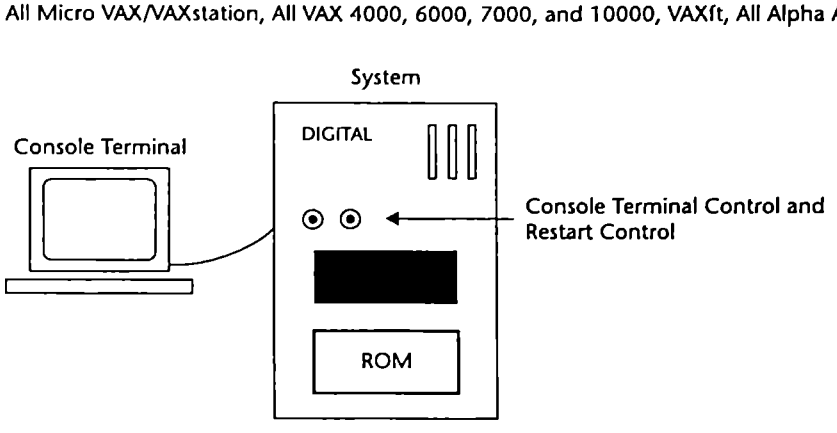
the necessary procedures. To maximize system uptime you should carefully review and familiarize yourself with these procedures, so that you are ready to use them when needed.

This chapter presents the steps necessary to prepare for each booting method, including setting hardware console switches, configuring console subsystem boot files, setting console parameters, and building stand-alone backup kits. Much of the information presented here is applicable to other processors not mentioned. For example, procedures for booting VAXserver systems are not specifically addressed; however, the procedures are similar to those for the corresponding VAX and MicroVAX systems.

4.3 Console Subsystems

Each VAX and Alpha AXP processor contains a console subsystem that consists of hardware and software mechanisms used to configure and control the booting process. When a system is powered on, the console subsystem loads a console program from a load device. Next, the console program either initiates a system boot or displays a console prompt and waits for manual console commands. Console commands perform limited functions such as loading central processing unit (CPU) registers, halting the main system processor, or

Figure 4.2 Systems with Nondedicated Console Subsystems



initiating a boot sequence. Each system model has either a *dedicated* or *nondedicated* console subsystem (Figures 4.1 and 4.2).

Processors with dedicated console subsystems typically include a console CPU (in addition to the CPU of the main system) and a console load device (usually a floppy drive or small hard disk). Boot procedures and console initialization code are usually loaded from the console load device. You configure boot procedures by modifying boot command files (e.g., DEFBOO.COM or DEFBOO.CMD) located on the console load device.

On processors with nondedicated console subsystems, all console functions are performed by the main CPU of the system. Console initialization code is typically read from read-only memory (ROM). Because these systems have no secondary storage device, boot parameters are stored in nonvolatile memory, rather than in boot command files.

4.3.1 Console Terminal

All console functions are performed from the console terminal, which is generally a conventional VT-style video terminal or LA-style hard-copy terminal. Most processors support two console modes: *console input/output* (CIO) mode or *program input/output* (PIO) mode. These two modes are henceforth referred to as *console mode* and *program mode*.

In program mode, the console terminal acts like any other user terminal attached to the system. If OpenVMS is already booted, pressing the <RETURN> key initiates an interactive login—the standard "Username:"

prompt is displayed, allowing you to log in and run any OpenVMS program or utility.

In console mode, the console mode prompt (usually >>>) is displayed, allowing console commands to be entered and processed by the console subsystem. You cannot enter OpenVMS commands from console mode.

You switch from *program mode* to *console mode* by entering a special key sequence from the console terminal. This switch sequence is either a CTRL/P or the <BREAK> key (sometimes labeled F5), depending on the CPU type. On some processors, you switch from program to console mode by pressing the HALT switch located on the system chassis.

You switch from *console mode* to *program mode* by using one of the following commands:

```
>>> CONTINUE                    ! Systems with nondedicated consoles

or

>>> SET TERMINAL PROGRAM        ! Systems with dedicated consoles
```

4.3.2 Console Terminal Control

Each system has a *console terminal control* mechanism that enables and disables the ability to switch the console terminal from program mode to console mode. Normally, terminal control is set with a physical switch located on the system chassis.

You should configure the console terminal control mechanism of your system to disable console-mode switching, except when absolutely necessary. On systems with nondedicated console subsystems, switching into console mode halts the main processor, forcing suspension of all system activity. Although you can usually resume system activity by issuing the CONTINUE command, you can easily crash the system or cause a reboot by accidentally entering the wrong command (e.g., BOOT). In addition, if the system is a member of a VMScluster and it is left in console mode for longer than the period specified by the system parameter reconnect interval (RECNXINTERVAL, usually 20 seconds), the cluster connection manager ejects one or more systems from the cluster. This occurs because the halted system cannot respond to periodic polling as required.

On some MicroVAX systems, you have to be especially careful to ensure that terminal control is disabled. If console-mode switching is enabled and

Table 4.1 Restart Control Settings

Restart Control Setting	*Boot Action*
BOOT	When restart control is set to BOOT the system initiates an automatic boot after power-on or system crash. Automatic boot procedures are discussed in Section 4.5.
RESTART/BOOT	If restart control is set to RESTART/BOOT and a power failure recovery occurs, the system attempts a system restart. A restart attempts to restart the CPU at the point just before the power failure. In many cases, user activity resumes at the point it was interrupted, and a system boot is not necessary. This functionality exists only for systems that have main memory battery backup. If the restart attempt fails, an automatic boot is performed.
RESTART/HALT	If restart control is set to RESTART/HALT, the system initiates a restart attempt after recovery from a power failure. However, if the restart fails, the system halts and displays the CIO mode prompt (>>>).
HALT	When restart control is set to HALT the system halts and displays the CIO mode prompt (>>>) after power-on or a system crash.

you power the console terminal off and on, a <BREAK> signal is generated that forces the console terminal into console mode, suspending all user activity and resulting in eventual cluster ejection.

When you enter console mode on a system with a dedicated console, the main processor is not halted. Thus, normal system activity is not affected. Nor is the system susceptible to cluster ejection unless a console HALT command is explicitly executed. As a rule, you should still disable console-mode switching on all system consoles to prevent accidental system halts or reboots.

The terminal control mechanisms for each processor are detailed in Appendix C.

4.3.3 System Restart Control

The *restart control* mechanism controls the boot action a system takes when the system is initially powered on. Table 4.1 illustrates each restart control setting and its associated boot action.

In most cases, you'll want to configure restart control so that system availability is automatically restored. If your system has memory battery backup, set restart control to RESTART/BOOT; otherwise set it to BOOT.

If you have applications that require manual recovery procedures after a system crash or power failure, you may need to set restart control to either RESTART/HALT or HALT (depending on whether or not you have battery backup). When either HALT option is selected, the power-on action is to halt and display the console mode prompt (>>>). This allows you to manually boot the system and temporarily prevent user access while you perform recovery procedures.

You should disable automatic booting when intermittent hardware failures begin to occur. By forcing the system to halt, your hardware service vendor can diagnose hardware problems because the state of the system immediately following the failure is preserved. This technique is most useful when hardware problems are so severe that a crash dump is not created.

The specific terminal control mechanisms for each processor are detailed in Appendix C.

4.4 Configuring a Boot Device

OpenVMS exists as a series of executable image files that reside on a system disk. Before OpenVMS is loaded, the system can perform only extremely limited functions, such as loading general CPU registers and reading files from disk into memory. Booting essentially involves three steps: locating a valid system disk, loading OpenVMS image files into memory, and transferring control to the operating system.

Most boot configuration steps involve specifying the OpenVMS system disk from which you want to boot. For example, you might want to boot from one of the following sources:

- Local disk
- Local tape (for stand-alone backup)
- HSC-attached disk (Hierarchical Storage Controller) via a CI
- DSSI-attached disk (Digital Storage System Interconnect)
- Network interface to an Infoserver compact disk–read-only memory (CD-ROM)
- Network interface to a Local Area VMScluster (LAVC) boot server

Table 4.2 Boot Control Flags (Register R5)

Bit Position	*Purpose*
<31:28>	System root (0–F)
<27: 1>	Not applicable
<0: 0>	Conversational boot (0 = No, 1 = Yes)

In addition, within a VMScluster, a system disk contains multiple *system roots*—one for each member of the cluster (e.g., [SYS0], [SYS1]). Each root contains the configuration files and startup procedures specific to its associated cluster member. To boot a system into a cluster, you must configure boot parameters to locate a valid system disk *and* specify the appropriate system root.

4.4.1 Processors with Nondedicated Console Subsystems

On processors with nondedicated console subsystems, you configure a default boot disk and system root by establishing values for the following nonvolatile boot registers:

Default boot device
Default boot control flags

For example, on a MicroVAX 3900, you configure these registers using the following console commands:

```
>>> SET BOOT device              ! Set default boot device
>>> SET BFLG boot-flags         ! Set default boot control flags
```

Boot control flags must be specified as a 32-bit hexadecimal number. As illustrated in Table 4.2, the upper four bits specify the system root (0–F) and the lower bit is used to select a conversational boot (discussed later in this chapter).

Thus, to configure the above system to boot from the device DUA1 and system root [SYS1], you issue the following console commands:

```
>>> SET BOOT DUA1
>>> SET BFLG 10000000
```

The console commands required to configure the default boot device and default boot control flags for other processors are detailed in Appendix C.

4.4.2 Systems with Dedicated Console Subsystems

On processors with dedicated console subsystems, the default boot device and boot control flags are configured with boot command procedures located on the console load device. When a system in this category boots, it executes the boot procedure DEFBOO.COM or DEFBOO.CMD.

The commands within the DEFBOO load processor register R0–R4 to specify the path to a valid system disk and load register R5 to indicate the desired system root (see Table 4.2).

Specifically, the function of each processor register is as follows:

```
R0   Boot device type code
R1   Boot device bus address
R2   Boot device controller information
R3   Boot device unit number
R4   Logical block number (LBN) of boot block on disk ! Usually not used
R5   Boot control flags
```

When booting from a locally attached disk, registers R0–R2 specify the local disk controller. The required values depend on the type of controller (e.g., Massbus, Unibus, BI). To determine the required values, review the template boot procedures for each controller type included on the console device of your system. The tables in Appendix C list the template procedures for each processor.

For example, boot procedure DU0BOO.CMD exists on the VAX 8600 console disk and contains the following commands:

```
! DU0BOO.CMD - Boot from UDA50-attached disk
!
DEPOSIT R0      11
DEPOSIT R1      5
DEPOSIT R2      0103F468
DEPOSIT R3      disk unit #
```

To boot from device DUA1, system root [SYS9], create a DEFBOO.CMD that uses the same values for R0–R2, but substitute appropriate values for R3 and R5, as follows:

```
! DEFBOO.CMD - Default boot (copied from DU0BOO.CMD)
!
DEPOSIT R0      11
DEPOSIT R1      5
DEPOSIT R2      0103F468
DEPOSIT R3     1
DEPOSIT R5      90000000
```

When booting from an HSC-attached disk, registers R0 and R1 specify the CI controller, and R2 specifies the HSC node number. As indicated in Appendix C, the template boot procedure to boot from the CI from a VAX 8600 is as follows:

```
! CIBOO.CMD - Boot from HSC-attached disk
!
DEPOSIT R0      20
DEPOSIT R1      E
DEPOSIT R2      HSC node #
DEPOSIT R3     disk unit #
```

Again, to boot from the device DUA1, attached to HSC node #3, system root [SYS9], you create a DEFBOO.CMD that uses the same values for R0–R1 but substitutes appropriate values for R2, R3, and R5, as follows:

```
! CIBOO.CMD - Boot from HSC-attached disk
!
DEPOSIT R0      20
DEPOSIT R1      E
DEPOSIT R2      03
DEPOSIT R3     1
DEPOSIT R5     90000000
```

If HSC-attached disks are connected to two HSCs, you should adjust register R2 to include both nodes. For example, to specify HSC nodes #3 and #4, include the following command in your boot procedure:

```
DEPOSIT R2      0304
```

4.4.3 Editing Boot Command Files

Boot command files are stored on the console device and are not directly accessible from OpenVMS. The procedures for modifying boot command files is processor-specific, as follows.

VAX 11/780/785, VAX 8600/8650, VAX 8200/8300/8250/8350

Use the OpenVMS Exchange Utility to copy boot command files from the console device to OpenVMS, modify them with any OpenVMS editor, then copy them back to the console device, as follows:

```
$ RUN SYS$SYSTEM:SYSGEN              ! Make CSA1 available
SYSGEN> CONNECT CONSOLE
$
$ EXCHANGE                           ! Invoke Exchange
EXCHANGE> COPY CSA1:DEFBOO.CMD *.*   ! Copy from console to OpenVMS
EXCHANGE> EXIT

$ EDIT DEFBOO.CMD                    ! Invoke OpenVMS editor
[make changes]

$ EXCHANGE
EXCHANGE> COPY DEFBOO.CMD CSA1:      ! Copy to console
```

VAX 8500/8530/8550/8700/8800

The Pro console subsystem has a DCL–like interface (PRODCL) that includes an editor similar to EDT (the OpenVMS text editor). To edit a boot file, issue the following console commands:

```
>>> QUIT                        ! Return to PRODCL
$                               ! PRODCL prompt (not OpenVMS DCL prompt)
$ EDIT/EDT filename             ! Run the editor
[make changes]
^Z

$ RUN CONTROL or
$ RUN CONSOLE                   ! Return to console prompt
```

VAX 8810/8820/8830/8840

The VAX 88x0 console subsystem consists of a MicroVAX II running OpenVMS. You edit boot files by accessing OpenVMS on the console and using the standard OpenVMS editor. This is accomplished using the following console commands:

```
PS-OS-0> QUIT
System Power is ON, are you sure?> Y
>                                    ! Console OpenVMS prompt
```

```
> EDIT DEFBOO.CMD
[make changes]
^Z
> POLARIS                              ! Return to console prompt
PS-OS-0>
```

4.4.4 Boot Sequence

Once a system boot is initiated, the following events occur

- General purpose CPU registers R0–R5 are loaded on the basis of either default values stored in nonvolatile random access memory (RAM) qualifiers appended to a BOOT command or as set by explicit DEPOSIT register commands in boot command files.
- The system disk is located on the basis of the contents of registers R0–R4. VMB.EXE is loaded from either console-attached media (disk or tape), from ROM, or from the OpenVMS system disk. VMB loads the secondary bootstrap program, SYSBOOT, from the file SYS$SYSTEM:SYSBOOT.EXE located on the system disk.
- SYSBOOT loads the system parameters from the file SYS$SYSTEM: VAXVMSSYS.PAR. Next, SYSBOOT checks the boot control flags to see if a conversational boot has been requested. If the low-order bit is set, the boot procedure stops and the SYSBOOT> prompt is displayed on the console terminal. Otherwise, SYSBOOT proceeds to load the OpenVMS executive, including SYS$SYSTEM:SYS.EXE, and transfers control to the routine EXE$INIT.
- EXE$INIT executes in system context, configuring key OpenVMS data structures, and initializing process scheduling and memory management. Next, the SWAPPER process is created, and process context is enabled.
- SWAPPER initiates system startup procedures (see chapter 5). Once startup procedures are complete, users can log in to the system.

4.5 Boot Methods

The following sections describe each of the most important booting methods and the conditions under which they are invoked. The procedures for configuring and invoking each are detailed in Appendix C.

Table 4.3 Automatic Boot Conditions

Condition	*Restart Control Setting*
Initial system power-on or power-failure recovery	BOOT or RESTART/BOOT (if restart attempt fails)
Fatal OpenVMS crash	BOOT or RESTART/BOOT
Shutdown requested automatic reboot	Don't care

4.5.1 Automatic Boot

The system automatically boots under the conditions listed in Table 4.3.

You must configure restart control to BOOT or RESTART/BOOT if you want systems to perform an automatic boot after initial power-on, power-failure recovery, or a system crash. However, an automatic boot is always initiated whenever one is optionally requested during an orderly system shutdown or after forcing a system crash directly with OPCCRASH. System shutdown and OPCCRASH are discussed in chapter 6.

4.5.2 Manual Boot

If a system does not perform an automatic boot (because of restart control settings), you must interactively enter boot commands from the console terminal and initiate a manual boot. These commands should specify which disk to boot from and set boot control flags as necessary.

In general, you use manual boot procedures to select alternative boot methods, such as:

- Default boot
- Conversational boot
- Stand-alone backup boot
- Alternative system disk boot

Before you can initiate a manual boot, you must switch the console terminal into console mode. This can be accomplished with one of the following techniques.

If OpenVMS is currently booted, perform an orderly system shutdown, but do *not* request an automatic reboot. After the shutdown is complete, enable

console-mode switching and enter the appropriate switch sequence (e.g., CTRL/P, <BREAK> key, or press the HALT button).

If the system is not currently powered on, enter console mode by first setting the processor's restart control mechanism to HALT or RESTART/HALT, then apply power. After power-on diagnostics are complete, the console mode prompt (>>>) is displayed.

4.5.3 Default Boot

You initiate a default boot by issuing the console BOOT command (without any qualifiers or parameters), as follows:

```
>>> BOOT
```

The default boot actions for each processor are described in Appendix C. On most systems, a default boot performs the same steps as an automatic boot. However, carefully review Appendix C because this is not always the case.

4.5.4 Conversational Boot

A conversational boot allows you to interactively adjust system parameters during the booting process by invoking the SYSBOOT utility before system startup procedures are initiated. You need to perform a conversational boot whenever any of the following conditions occur:

- All passwords to privileged accounts are lost
- Incorrect system parameter settings prevent booting
- Changes to system startup procedures prevent booting

The procedures for initiating a conversational boot for each processor are discussed in Appendix C. SYSBOOT is discussed in more detail in Section 4.6.

4.5.5 Stand-alone Backup Boot

Stand-alone backup is a utility used to perform disk backup and restore operations without OpenVMS. Stand-alone backup is actually a minimal version of the OpenVMS operating system. The conditions under which stand-alone backup is required are described in chapter 13. Before you can boot stand-alone backup you have to create a kit on a device that is accessible during

system boot. The procedures for creating Stand-alone Backup Kits are described in Section 4.8.

When you build a Stand-alone Backup Kit on a system disk it is built into the [SYSE] system root. Therefore, to boot the kit, you must load the upper four bits of the boot control flags (register R5) with the hexadecimal value E. For example, on MicroVAX systems this is accomplished as follows:

```
>>> B/E0000000 disk-device
>>> B/E0000000 DUA0 (for, example)
```

On the other hand, when you build a Stand-alone Backup Kit on a nonsystem disk, it is built into the [SYS0] system root. Therefore, you specify a value of 0 for the upper four bits of the boot control flags (register R5). On microVAX systems, zero is the default value for the boot control flags, so you can simply enter the standard boot command followed by the name of the device containing the Stand-alone Backup Kit. For example:

```
>>> B disk-device
>>> B DUA1 (for, example)
>>> B tape-device
>>> B MUA0 (for, example)
```

The specific procedures for booting stand-alone backup for each processor are presented in Appendix C.

4.5.6 Alternative System-disk Boot

If you have spare disks, you should consider establishing an alternative system disk. You can create an alternative system disk either by making an exact copy of your current system disk using OpenVMS backup or by installing a minimal OpenVMS kit using the VMSKITBLD command procedure. With an alternative system disk, you can restore minimal system availability in the event your primary system disk fails. This might allow some users temporary access to critical applications. At a minimum, you have the ability to boot the system, which can be a big help in troubleshooting problems related to the primary system disk.

The procedures for creating an alternative system disk are discussed in Section 4.9.

4.6 SYSBOOT Options

Before you use the following procedures you must perform a conversational boot using the processor-specific procedure described in Appendix C. Once you initiate a conversational boot, the booting process invokes the SYSBOOT utility just before executing system startup procedures. You can then enter a series of commands at the SYSBOOT prompt as follows:

```
SYSBOOT>
```

When you have finished entering commands, enter CONTINUE so that the boot process will proceed.

From SYSBOOT you can perform the following functions:

- Perform minimal system startup (prevent execution of SYSTART-UP_V5.COM)
- Bypass system startup (prevent execution of STARTUP.COM)
- Select alternative user authorization file (UAF)
- Adjust system parameters

Most of these functions involve changing system parameters. By default parameters changed with SYSBOOT are permanently written to the system parameter file (e.g., VAXVMSSYS.PAR). If you want changes to be temporary, you should set the parameter WRITESYSPAR to 0. This prevents changes from being written to the system parameter file.

4.6.1 Minimal Startup

When performing system maintenance, or when boot problems occur, you may want to perform a minimal startup. A minimal startup prevents the main site-specific startup command file, SYS$MANAGER:SYSTARTUP_Vxx, from being executed. This can be useful if you don't want the full system activated or want to save time during reboots. You may also need to perform a minimal startup prior to installing an OpenVMS upgrade.

To initiate a minimal startup, respond to the SYSBOOT prompt as follows:

```
SYSBOOT> SET STARTUP_P1 "MIN"
SYSBOOT> CONTINUE

.... System Boots, nonsystem logins are disabled
```

```
Perform login:

Username:
Password:
```

You must manually reset STARTUP_P1 as follows before the next system reboot:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> SET STARTUP_P1 " "
```

4.6.2 Bypass Startup

If you experience system startup problems that extend beyond SYSTARTUP_Vxx.COM, you can bypass the execution of all startup procedures:

```
SYSBOOT> SET/STARTUP OPA0:
SYSBOOT> CONTINUE
... System Boots
$                    ! DCL prompt displayed
$ SET NOON           ! disable default error handler
$
```

After the system boots, the DCL prompt ($) is displayed, from which you can enter commands to research the problem. You must first enter the SET NOON command to disable the default DCL error handler. Otherwise, if you enter a command that results in a severe or fatal error, your process is terminated and you have to reboot the system. When you have finished debugging, you can invoke normal system startup procedures as follows:

```
$ @SYS$SYSTEM:STARTUP
```

4.6.3 Bypass Password Checking

If you lose passwords to all privileged accounts, you can force OpenVMS to switch to the alternative UAF as follows:

```
SYSBOOT> SET UAFALTERNATE 1
SYSBOOT> CONTINUE
...allow boot to complete
```

By default, OpenVMS uses the logical name SYSUAF to locate the UAF file. If the logical name is undefined, then OpenVMS uses the file SYS$SYS-

TEM:SYSUAF.DAT. When UAFALTERNATE is set, OpenVMS defines the logical name SYSUAF as follows:

```
$ DEFINE/SYSTEM/EXEC SYSUAF SYSUAFALT
```

OpenVMS then looks for the file SYS$SYSTEM:SYSUAFALT.DAT to use as the UAF; however, this file does not normally exist. When the UAF file does not exist, any username and password combination entered from the console terminal (OPA0: only) allows a successful login. No matter what username you specify, you will be given full system privileges. For example:

```
Username: BOGUS (From console terminal)
Password: type anything
Password: type anything

$ DEFINE/SYSTEM/EXEC SYSUAF SYS$SYSTEM:SYSUAF.DAT
(switch back to normal UAF)

$ RUN AUTHORIZE
UAF> MODIFY SYSTEM/PASSWORD=new-password
UAF> EXIT

$ RUN SYSGEN
SYSGEN> USE CURRENT
SYSGEN> SET UAFALTERNATE 0
SYSGEN> WRITE CURRENT
SYSGEN> EXIT

(Users can now log in)
```

After you log in, you should redefine the SYSUAF logical name to point to your normal SYSUAF.DAT file. Then use AUTHORIZE to change passwords as needed. Finally, invoke SYSGEN and reset UAFALTERNATE to zero (0), otherwise, the next time you boot the system, the alternative UAF will be selected.

4.6.4 Adjust SYSGEN Parameters

You can use SYSBOOT to change any system parameter before booting as follows:

```
SYSBOOT> SET SRPCOUNT 1000
SYSBOOT> SET BALSETCNT 150
SYSBOOT> CONTINUE
```

You should use this mechanism only in an emergency. Otherwise, change system parameters with the AUTOGEN utility as discussed in chapter 18.

4.6.5 Reverting to OLD Parameters

When you change system parameters with AUTOGEN, the previous settings are saved in the file VAXVMSSYS.OLD. If the system does not boot because of a system parameter problem, you can return to the previous parameter settings as follows:

```
SYSBOOT> USE VAXVMSSYS.OLD
SYSBOOT> CONTINUE
```

4.6.6 Reverting to DEFAULT Parameters

If returning to previous parameters does not allow you to boot, you can revert to DEFAULT parameters as follows:

```
SYSBOOT> USE DEFAULT
SYSBOOT> CONTINUE
```

The default parameters are designed to allow any system to boot. However, you may not want to select the default values for all parameters. For instance, VMScluster parameters such as SCSNODE default to blank spaces. If you want to boot as a VMScluster member, you must first manually set these parameters to their proper values.

4.7 Interrupt Priority C

If a cluster hangs because of loss of quorum, you can manually restore quorum by invoking the IPC utility as follows:

```
Enter CIO mode on console terminal with CTRL/P or <BREAK>
>>>
>>> HALT
>>> D/I 14 C          ! Set Interrupt Priority Level 12
>>> CONTINUE
IPC> Q                ! recalculate quorum
IPC> ^Z               ! exit IPC
```

The IPC Q command bypasses the standard quorum calculation formula and restores quorum by setting its value equal to the current number of votes present.

WARNING: IPC is extremely dangerous. Quorum is meant to prevent a single cluster from partitioning into multiple unsynchronized clusters. If a cluster loses quorum because of a hardware failure, *do not* restore quorum with IPC—cluster accessible disks may become corrupted. You should only use IPC when you know quorum was lost because one or more members shut down.

4.8 Building Stand-alone Backup Kits

If your system disk fails, you must restore backups using the Stand-alone Backup Utility. Stand-alone backup is invoked by issuing special boot commands. Instead of booting OpenVMS, you boot stand-alone backup from the system disk, an alternative disk, the console device, or tape. Before you can boot from any of these devices, you must create a Stand-alone Backup Kit on the device using STABACKIT as follows:

```
$ @SYS$UPDATE:STABACKIT
```

When you build a Stand-alone Kit on an existing system disk, an additional system root [SYSE] is created. The files necessary to boot stand-alone backup are created in the SYSE root. The build procedure actually uses alias directory entries to point to existing system directories, so there is really only one physical copy of each file. For this reason, building the kit on a system disk uses very little disk space.

When you build a Stand-Alone Kit on a nonsystem disk, a [SYS0] root directory is created. Because there are no operating system files on this disk, they must be physically copied from the system disk to the alternative disk. This does use more disk space than when you build kits on a system disk; however, you should build at least two kits in case one fails. As an additional backup, you should build kits on tape and console media (if applicable). However, under normal circumstances, boot stand-alone backup from disk. Booting from other devices can take 10–15 minutes longer.

4.9 Creating Alternative System Disks

Before you can boot OpenVMS from an alternative disk, you must make a copy of your existing system disk with either VMSKITBLD or OpenVMS backup as follows.

Invoke VMSKITBLD with the following command:

```
$ @SYS$UPDATE:VMSKITBLD
```

The procedures for using VMSKITBLD are documented in *Setting Up an OpenVMS System.*

You can also use the OpenVMS Backup Utility to perform an image backup of your primary system disk to an alternative system disk as follows:

```
$ DISMOUNT/NOUNLOAD DUA1:              ! Mount alternate disk
$ MOUNT/FOREIGN DUA1:                  ! Target must be /FOREIGN
$ BACKUP/IMAGE/IGNORE=INTERLOCK DUA0: DUA1:
```

Note that DUA1 is initialized, causing its previous contents to be erased.

4.10 Conclusion

Normally, booting OpenVMS is a simple process; however, you should understand *all* booting options. Otherwise, when emergencies occur you will waste precious minutes trying to figure out what to do. You should thoroughly test all booting methods so that you know they'll work when you need them.

4.11 Additional References

Detailed boot procedures are *not* contained in the standard OpenVMS manual set; they are included in the hardware manuals that came with your system. Some boot procedures are discussed in the Installation and Upgrade supplements that are included with OpenVMS media and documentation kits; however this documentation covers only the most recent VAX systems.

Digital Assisted Services US Edition Price List, catalog for hardware manuals, available from DECdirect.

VAX/VMS Internals and Data Structures (V5.2), chapter 30, "Bootstrap Procedures," Digital Press, 1991.

5 SYSTEM STARTUP PROCEDURES

5.1 Chapter Objectives

- Increase system availability by developing system startup procedures that are clear, concise, and follow commonly practiced standards
- Demonstrate techniques that reduce system boot time
- Discuss aspects of DCL error processing that are important to system startup procedures
- Provide techniques for automatically detecting and reporting system startup errors
- Provide a fully developed set of template startup procedures

5.2 Introduction

System startup is a continuation of the system initialization process initiated by a system boot. For the purpose of this book, system startup involves all system initialization steps that occur in the context of a process, rather than in system context. Chapter 4 discusses booting steps performed in system context.

Sometimes system startup procedures fail because of procedure bugs or hardware problems. By default, system startup errors and messages are displayed on the console terminal. Unless you read every line displayed on the console, you'll miss key messages indicating startup problems. You need a more reliable method of identifying startup errors.

System startup is an involved process—as many as 50 different command procedures are executed. To minimize downtime, you should thoroughly understand when, where, and why each procedure is executed. In addition,

many startup procedures depend on the successful completion of prior procedures. You should understand system startup dependencies so that you preserve the correct startup order when you make modifications.

5.3 System Startup Sequence

System startup involves the execution of two sets of command procedures:

- Site-independent startup procedures
- Site-specific startup procedures

Site-independent startup procedures are DCL command procedures supplied by Digital. These procedures are the same for every OpenVMS system and *should not be modified.* Any changes made to site-independent procedures will probably be overwritten during future OpenVMS upgrades. *Site-independent* procedures perform functions such as initializing OpenVMS data structures, loading portions of the OpenVMS executive, defining logical names, and opening primary page, swap, and dump files. At certain points within site-independent startup procedures, site-specific procedures are invoked.

Site-specific procedures perform functions that depend on your hardware configuration and the needs at your site. For example, site-specific startup procedures perform the following: mount disks, configure terminal ports, establish print and batch queues, start layered products, and start applications. Digital supplies template procedures, but you must customize them to reflect the configuration at your site.

The following sections detail how, where, and when startup procedures are executed, beginning with the SWAPINIT startup phase.

5.3.1 SWAPINIT Phase

The final step of the boot process (as described in chapter 4) is to create the SWAPPER detached process. SWAPPER is the first process created in the life of a OpenVMS system. During normal OpenVMS operation, SWAPPER performs memory management functions; however, during system startup, SWAPPER performs the following startup-related functions:

- Initializes paged pool
- Invokes OpenVMS executive initialization routines
- Initializes the logical name database

- Creates the SYSINIT detached process
- Branches to main SWAPPER loop and begins normal memory management functions

5.3.2 SYSINIT Phase

The SYSINIT process then performs the following startup functions:

- Loads portions of the OpenVMS executive
- Performs VMScluster initialization (if applicable)
- Opens primary page, swap, and dump files
- Activates Files-11 Extended QIO Processor (XQP)
- Loads system message files
- Defines key OpenVMS logical names (e.g., SYS$SYSROOT, SYSSCOMMON)
- Creates the STARTUP process

5.3.3 STARTUP Phase

The STARTUP process begins by executing the primary site-independent startup procedure as specified in the system parameter database. By default, this primary startup file is SYS$SYSTEM:STARTUP.COM. You can verify this by issuing the following commands:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> SHOW/STARTUP
Startup command file = SYS$SYSTEM:STARTUP.COM
```

When errors occur during the startup process, you may need to temporarily disable the execution of STARTUP.COM. This is accomplished by invoking a *conversational boot*, as discussed in chapter 4. STARTUP.COM subsequently invokes the remaining site-independent startup procedures as specified in the *startup database*. The startup database consists of three files contained in the SYS$STARTUP directory. They are described by the three logical names

```
STARTUP$PHASES
STARTUP$STARTUP_VMS
STARTUP$STARTUP_LAYERED
```

The following sections detail the contents of these files, which are the same for OpenVMS 5.5 and OpenVMS 6.0.

STARTUP$PHASES

The PHASES data file defines the names and order of the system startup phases:

```
INITIAL
DEVICES
PRECONFIG
CONFIG
BASEENVIRON
LPBEGIN
LPMAIN
LPBETA
END
```

STARTUP$STARTUP_VMS

STARTUP_VMS contains a list of site-independent command procedures that are to be executed by STARTUP.COM. Several of these procedures invoke the site-specific startup procedures located in the SYS$MANAGER directory. Table 5.1 lists the order in which site-independent command procedures are executed and indicates when and where site-specific startup procedures are invoked.

STARTUP$STARTUP_LAYERED

The logical name STARTUP$STARTUP_LAYERED locates the file that contains layered product startup procedures. By default, this file is empty, but when you install some Digital products with VMSINSTAL, the startup procedure for the product is automatically inserted in STARTUP_LAYERED. These entries usually are designated to be executed during the LPMAIN startup phase.

The STARTUP sequence begins by setting the current phase to the first phase listed in STARTUP$PHASES (ie, INITIAL). Then all command procedures listed in STARTUP$STARTUP_VMS with a phase that matches the current phase are executed. Next, all procedures in STARTUP$STARTUP_LAYERED that match the current phase are executed. Finally, the current phase is set to the next phase listed in STARTUP$PHASES, and the process repeats until the last phase is completed (i.e., END).

Table 5.1 Site-independent and Site-specific Startup Procedures

Startup Phase	*Site-independent Procedure (in SYS$STARTUP)*	*Site-specific Procedure (in SYS$MANAGER)*
INITIAL	VMS$INITIAL-050_VMS.COM VMS$INITIAL-050_LIB.COM VMS$INITIAL-050_CONFIGURE.COM	None
DEVICES	VMS$DEVICE_STARTUP.COM	==> SYCONFIG.COM ==> SYLOGICALS.COM ==> SYPAGSWPFILES.COM
PRECONFIG	IPC$STARTUP.COM	None
CONFIG	VMS$CONFIG-050_VMS.COM VMS$CONFIG-050_ERRFMT.COM VMS$CONFIG-050_CACHE_SERVER.COM VMS$CONFIG-050_CSP.COM VMS$CONFIG-050_OPCOM.COM VMS$CONFIG-050_AUDIT_SERVER.COM VMS$CONFIG-050_JOBCTL.COM VMS$CONFIG-050_LMF.COM VMS$CONFIG-050_SHADOW_SERVER.COM	==> SYSECURITY.COM
BASEENVIRON	VMS$BASEENVIRON-050_VMS.COM VMS$BASEENVIRON-050_SMISERVER.COM VMS$BASEENVIRON-050_LIB.COM LICENSE_CHECK.EXE DECDTM$STARTUP.COM	None
LPBEGIN	VMS$LPBEGIN-050_VMS.COM VMS$LPBEGIN-050_STARTUP.COM	==> SYSTARTUP_V5.COM or ==> SYSTARTUP_VMS.COM
LPMAIN	None	None
LPBETA	None	None
END	None	None

A standard structure for organizing site-specific startup procedures has never been specified; as a result, most system managers locate site-specific startup commands in SYSTARTUP_Vxx and related files. Digital is in the process of developing a more structured approach by allowing you to incorporate

Table 5.2 Site-specific Startup Procedures and Templates

Site-specific Startup File	*Digital-supplied Template File*
SYS$MANAGER:SYCONFIG.COM	SYS$MANAGER:SYCONFIG.TEMPLATE
SYS$MANAGER:SYLOGICALS.COM	SYS$MANAGER:SYLOGICALS.TEMPLATE
SYS$MANAGER:SYPAGSWPFILES.COM	SYS$MANAGER:SYPAGSWPFILES.TEMPLATE
SYS$MANAGER:SYSECURITY.COM	SYS$MANAGER:SYSECURITY.TEMPLATE
SYS$MANAGER:SYSTARTUP_Vxx.COM	SYS$MANAGER:SYSTARTUP_Vxx.TEMPLATE

startup procedures into the startup database. Even now, you can migrate site-specific startup procedures to the STARTUP_LAYERED database using the SYSMAN Utility (see the *OpenVMS System Manager's Utility Manual*). However, these mechanisms have not been fully developed and are subject to change and therefore are not discussed in this chapter.

5.4 Site-specific Startup Procedures

Table 5.2 lists the five site-specific startup procedures in the order in which they are invoked. When OpenVMS is initially installed, each site-specific startup file is identical to its respective Digital-supplied template file. The site-specific startup procedures must be customized to reflect the needs of the site. Retain the template files in their original form, because they may be required during future OpenVMS upgrades.

The following sections describe the purpose of each of the site-specific startup files and strategies for customizing them. Strategies for coordinating these startup procedures and system data files are discussed later in this chapter.

SYCONFIG.COM

SYCONFIG.COM configures hardware devices that are not automatically recognized by OpenVMS. If you have third-party equipment that requires manual configuration, SYCONFIG should contain commands listed in your hardware documentation; otherwise, SYCONFIG.COM should be empty.

SYLOGICALS.COM

The function of SYLOGICALS.COM is to define site-specific system logical names. The SYLOGICALS.COM supplied by Digital includes code that de-

Table 5.3 Logical Names Defined in SYLOGICALS.COM

Purpose	*Logical Name*	*Related Chapter*
System announcement messages	SYS$ANNOUNCE	11
	SYS$WELCOME	11
System login command procedure	SYS$SYLOGIN	11
Default system print and batch queue	SYS$BATCH	8
	SYS$PRINT	8
Cluster-common files	SYSUAF	9
	NETPROXY	9
	RIGHTSLIST	9
	VMSMAIL_PROFILE	9
	QMAN$MASTER	8
	NETNODE_REMOTE	8
	NETSERVER$TIMEOUT	8
OPCOM logical names	OPC$LOGFILES_CLASSES	15
	OPC$LOGFILES_ENABLE	15
	OPC$LOGFILE_NAME15	
	OPC$OPA0_CLASSES	15
	OPC$OPA0_ENABLE	15
SHUTDOWN	SHUTDOWN$INFORM_NODES	6
	SHUTDOWN$MINIMUM_MINUTES	6
	SHUTDOWN$DISABLE_AUTOSTART	6
Other data files	ACCOUNTNG (Accounting Data File)	

fines system logical names appropriate for smaller, stand-alone systems. For example, logical names $DISK1 and $DISK2 are defined as the first and second disk drives on the system. This book suggests logical names that are more appropriate for larger systems. If you wish to use the conventions in this book, start with an empty SYLOGICALS startup file instead of using Digital's template file. Table 5.3 lists the logical names you may want to define in SYLOGICALS.COM.

By convention, disks are usually mounted in SYSTARTUP_Vxx.COM. However, if you need certain disks mounted early in the startup process, you may want to mount them in SYLOGICALS.

Table 5.4 Modularized SYSTARTUP_Vxx.COM

SYSTARTUP_Vxx Function	*Invokes*
Mount disks	STARTUP_MOUNT
Start DECnet and local area transport	STARTUP_NETWORK
Configure devices and terminals	STARTUP_SETTERM
Initialize print and batch queues	STARTUP_QUEUES
Install images shareable	STARTUP_SHARE
Analyze crash-dump file	STARTUP_SDA
Start layered products	STARTUP_PRODUCTS
Start applications	STARTUP_APPLICATIONS
Define interactive login limit	Executed within SYSTARTUP
Notify users of system availability	Executed within SYSTARTUP

SYPAGSWPFILES.COM

The purpose of SYPAGSWPFILES is to install *secondary* page and swap files. This is in contrast to *primary* page and swap files installed during site-independent startup procedures. Chapter 18 discusses page and swap files in more detail. If you haven't already mounted the disk containing your secondary page files, you should mount it in SYPAGSWPFILES.

SYSECURITY.COM

The function of SYSECURITY.COM is to configure the security events you want to audit. Default security event auditing is automatically enabled by site-independent startup procedures, and all events are journaled to SYS$MANAGER:SECURITY_AUDIT.AUDIT$JOURNAL. Beginning with OpenVMS 6.0, the audit journal file was renamed to SECURITY.AUDIT$JOURNAL.

Chapter 17 discusses additional security events you may want to audit. Enable them by adding appropriate commands to SYSECURITY. If your system has a high volume of security events, causing excessive disk activity on your system disk, you can specify an alternative location for the audit journal file.

SYSTARTUP_Vxx.COM

Most site-specific startup tasks are contained in the file SYSTARTUP_Vxx. On systems running OpenVMS 5.x the name of this command procedure is SYSTARTUP_V5.COM. Beginning with OpenVMS 6.0, this file was renamed to

SYSTARTUP_VMS.COM. Throughout this chapter this procedure is referred to as *SYSTARTUP_Vxx.COM*.

If your site is large or complex, SYSTARTUP_Vxx.COM may contain hundreds of DCL commands, making it difficult to follow. Table 5.4 lists each of the important tasks performed in SYSTARTUP_Vxx. Whenever a task requires more than a few DCL commands, move them from SYSTARTUP_Vxx into a separate command procedure and include a line in SYSTARTUP_Vxx that invokes the new command procedure.

For example, if you need to mount more than one or two disks, put all your mount commands in a file such as STARTUP_MOUNT.COM, for example

```
$! STARTUP_MOUNT.COM
$!
$ MOUNT...
$ MOUNT...
.
.
```

Then configure SYSTARTUP_Vxx to invoke STARTUP_MOUNT as follows:

```
$! SYSTARTUP_Vxx.COM
$!
$ @SYS$MANAGER:STARTUP_MOUNT              ! Mount Disks
.
```

Note that the names of the command procedures in Table 5.4 all begin with "STARTUP_". This allows you to quickly locate startup-related command procedures by issuing the following command:

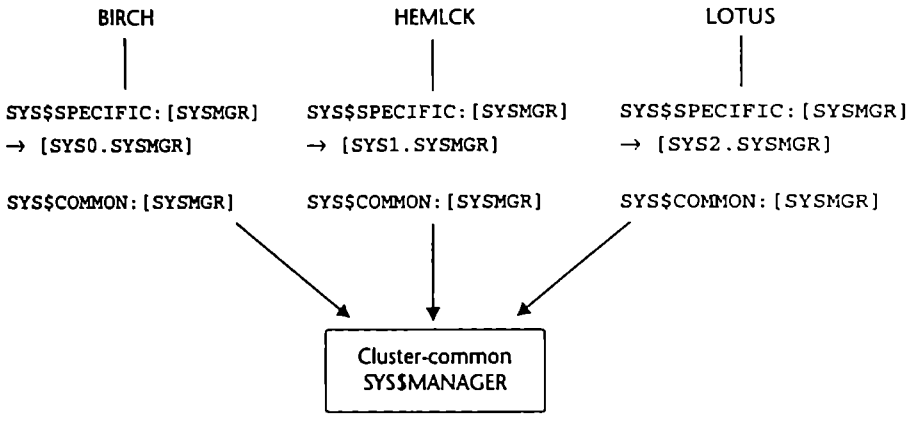
```
$ DIRECTORY SYS$MANAGER:STARTUP*.COM
```

The source code for the startup files listed in this chapter are located on the diskette included with this book (see Appendix E).

5.5 Coordinating Site-specific Startup Procedures

You must give special consideration to coordinating system startup procedures within a VMScluster environment. Tightly integrated clusters (i.e., *common-environment*) generally share a single system disk, offer the same applica-

Figure 5.1 OpenVMS System Disk Organization



tions, start the same print and batch queues, and for all intents and purposes appear to the user as a single system. On the other hand, less integrated clusters that utilize multiple system disks or offer different applications on each cluster member are referred to as *multiple-environment* clusters. Because members within a common-environment cluster offer a similar environment, you can generally maintain a single set of startup procedures. Within a multiple-environment cluster you may need to maintain a separate set of startup procedures for each system.

Regardless of whether a system is a VMScluster member, its system disk has a cluster structure—that is, it has a system-specific root and a cluster-common root. All startup procedures are located by the SYS$MANAGER logical name; however, SYS$MANAGER is defined in terms of the search list SYS$SYSROOT, as follows:

```
SYS$MANAGER
     |
     ==> SYS$SYSROOT:[SYSMGR]
            |
            | ==> SYS$SPECIFIC:[SYSMGR]
            |
            | ==> SYS$COMMON:[SYSMGR]
```

The structure of an OpenVMS system disk and system logical names definitions are discussed in more detail in chapter 3.

Figure 5.1 illustrates the SYS$MANAGER organization for a three node common-environment cluster booting from a common system disk. Chapter 3 discusses OpenVMS system disk organization in more detail.

Table 5.5 Startup File Organization for a Stand-alone System

Member	*BIRCH*
System disk	1DUA0:
Directory	SYS$COMMON:[SYSMGR]
System startup files	SYCONFIG SYLOGICALS SYPAGSWPFILES SYSECURITY SYSTARTUP_Vxx etc.

Depending on the size and complexity of your configuration, you locate system startup files in either the system-specific or cluster-common root. The following sections suggest specific strategies for typical environments.

5.5.1 Stand-alone System

On a stand-alone system, you need only one set of startup files, which you should locate in SYS$COMMON:[SYSMGR], as illustrated by Table 5.5. Locating startup files in the common root makes it easier to convert the system to a cluster.

5.5.2 Common-environment VMScluster (Simple)

If you maintain a simple common-environment cluster in which there are not many differences between the environments offered by cluster members, you can also maintain a single set of startup procedures in the common root, as illustrated in Table 5.6.

However, even in a common-environment cluster, some aspects of system startup are always system-specific (e.g., each system must have dedicated page and swap files). To handle these situations, add DCL IF-THEN statements for conditional execution of system-specific procedures. For example, you might modify SYPAGSWPFILES.COM as follows:

```
$!
$! SYPAGSWPFILES.COM
$!
$ node = f$edit(f$getsyi("SCSNODE"),"COMPRESS")    ! Get rid of spaces
```

Table 5.6 System Startup Procedures for a Simple Common-Environment Cluster

Member	*BIRCH*	*LOTUS*	*HEMLCK*	*ALL*
System Disk	1DUA0:	1DUA0:	1DUA0:	1DUA0:
Directory	SYS$SPECIFIC: [SYSMGR]	SYS$SPECIFIC: [SYSMGR]	SYS$SPECIFIC: [SYSMGR]	SYS$COMMON: [SYSMGR]
System Startup Files	None	None	None	SYCONFIG SYLOGICALS SYPAGSWPFILES SYSECURITY SYSTARTUP_Vxx ...

```
$!
$ if node .eqs. "BIRCH)"
$ then
$! ... Install BIRCH page files
$ endif
$!
$ if node .eqs. "LOTUS"
$ then
$! ... Install LOTUS page files
$ endif
$! etc.
```

5.5.3 Common-environment VMScluster (Complex)

In a common-environment cluster with many systems, adding DCL conditional statements for each system might be too cumbersome. In these situations, you can establish a system-specific startup procedure for each system (in the SYS$SPECIFIC:[SYSMGR] root) and invoke it from the common procedure (Table 5.7).

Note that the node name for each system is appended to the system-specific procedures (e.g., SYLOGICALS_BIRCH). This reduces the chance of confusing the startup procedures for one system with those of another and allows you to more easily invoke each procedure, for example:

Table 5.7 System Startup Procedures for a Complex Common-environment Cluster

Member	*BIRCH*	*LOTUS*	*HEMLCK*	*COMMON*
System disk	1DUA0:	1DUA0:	1DUA0:	1DUA0:
Directory	SYS$SPECIFIC: [SYSMGR]	SYS$SPECIFIC: [SYSMGR]	SYS$SPECIFIC: [SYSMGR]	SYS$COMMON: [SYSMGR]
System startup files	SYCONFIG _BIRCH	SYCONFIG _LOTUS	SYSCONFIG _HEMLCK	SYSCONFIG
	SYLOGICALS _BIRCH	SYSLOGICALS _LOTUS	SYSLOGICALS _HEMLCK	SYSLOGICALS
	SYPAGSWPFILES _BIRCH	SYPAGSWPFILES _LOTUS	SYSPAGSWPFILES _HEMLCK	SYPAGSWPFILES
	SYSECURITY _BIRCH	SYSECURITY _LOTUS	SYSECURITY _HEMLCK	SYSECURITY
	SYSTARTUP _BIRCH ...	SYSTARTUP _LOTUS ...	SYSTARTUP _HEMLCK ...	SYSTARTUP_Vxx ...

```
$!
$! SYLOGICALS.COM    (Cluster Common)
$!
$!
$ node = f$edit(f$getsyi("SCSNODE"),"COMPRESS")    ! Get rid of spaces
$!
$! Execute system-specific SYLOGICALS
$!
$ @SYS$MANAGER:SYLOGICALS_'node'
$!
$! Define cluster-common logicals
$!
$ DEFINE/SYSTEM ....
```

Do not automatically create a system-specific file for every startup procedure—you will quickly have dozens of command procedures and they will be difficult to manage. You should create system-specific procedures only when absolutely necessary.

Table 5.8 System Startup Procedures for a Multiple-environment Cluster

Member	*BIRCH*	*LOTUS*	*HEMLCK*
System disk	1DUA0:	1DUA1:	1DUA2:
Directory	SYS$COMMON: [SYSMGR]	SYS$COMMON: [SYSMGR]	SYS$COMMON: [SYSMGR]
System startup files	SYCONFIG SYLOGICALS SYPAGSWPFILES SYSECURITY SYSTARTUP_Vxx ...	SYCONFIG SYLOGICALS SYPAGSWPFILES SYSECURITY SYSTARTUP_Vxx	SYCONFIG SYLOGICALS SYPAGSWPFILES SYSECURITY SYSTARTUP_Vxx ...

5.5.4 Multiple-environment VMScluster

In a multiple-environment cluster in which each system boots from a different system disk and offers very different environments, you can use the same strategy as you would for a stand-alone system, except you maintain multiple sets of startup procedures (one for each cluster member). For example, in a three-member cluster (BIRCH, LOTUS, and HEMLCK), you might organize system startup files as show in Table 5.8.

If you have a multiple-environment cluster with a common system disk, you can use the same strategy as above, except locate all startup files in SYS$SPECIFIC:[SYSMGR] rather than SYS$COMMON.

5.6 Coordinating Shareable System Files

Within a VMScluster environment you must coordinate system data files to reflect the level of cluster integration. In general, within a common-environment cluster, all members share a single copy of data files such as the user authorization files, queue database files, and shareable DECnet database files. In a multiple-environment cluster, you usually maintain a separate copy of these files for each cluster member. Table 5.9 lists the system files that can be shared.

To share files among cluster members, you should first ensure that each file is located in the common root of its associated directory. For example,

Table 5.9 VMScluster Shareable Data Files

Shareable File	*Function*
SYS$SYSTEM:	
NETPROXY.DAT	DECnet proxy database
RIGHTSLIST.DAT	System rights identifiers
SYSUAF.DAT	System user authorize file (UAF)
SYSUAFALT.DAT	Alternative UAF
SYSALF.DAT	Automatic login Facility database
VMS$OBJECTS.DAT	Security object database (V6.0)
VMS$PASSWORD_HISTORY.DATA	Password history file
VMSMAIL_PROFILE.DATA	OpenVMS mail profiles
NETNODE_REMOTE.DAT	DECnet remote node database
NETOBJECT.DAT	DECnet object database
SYS$QUEUE_MANAGER.QMAN$JOURNAL	Queue journal file
SYS$QUEUE_MANAGER.QMAN$QUEUES	Queue database file
QMAN$MASTER.DAT	Queue master file
SYS$MANAGER:	
AUDIT_SERVER.DAT	Audit server database (V5.0)
VMS$AUDIT_SERVER.DAT	Audit server database (V6.0)
NETNODE_UPDATE.COM	Local area cluster database
SYS$LIBRARY:	
VMS$PASSWORD_DICTIONARY.DATA	Password dictionary

ensure that SYSUAF.DAT is located in SYS$COMMON:[SYSEXE]. Next, during system startup (in SYLOGICALS.COM) you should define system-wide logical names specifying the location of each shared file. For example, to share all user authorization files, add the following commands to SYLOGICALS:

```
$!
$! Logicals for System Data Files
$!
$ DSE:=DEFINE/SYSTEM/EXECUTIVE
$!
$ DSE NETPROXY SYS$COMMON:[SYSEXE]NETPROXY.DAT
$ DSE RIGHTSLIST SYS$COMMON:[SYSEXE]RIGHTSLIST.DAT
$ DSE SYSUAF SYS$COMMON:[SYSEXE]SYSUAF.DAT
$ DSE SYSUAFALT SYS$COMMON:[SYSEXE]SYSUAFALT.DAT        ! Not used here
$ DSE SYSALF SYS$COMMON:[SYSEXE]SYSALF.DAT              ! Not used here
$ DSE VMS$PASSWORD_HISTORY SYS$COMMON:[SYSEXE]VMS$PASSWORD_HISTORY.DATA
$ DSE VMS$PASSWORD_DICTIONARY SYS$COMMON:[SYSLIB]VMS$PASSWORD_DICTIONARY.DATA
$ DSE VMSMAIL_PROFILE SYS$COMMON:[SYSEXE]VMSMAIL_PROFILE
```

On the other hand, to maintain a separate copy of each of the above files, locate the files in the SYS$SPECIFIC root of each directory and define the logical names in the SYLOGICALS.COM startup file for each cluster member as follows:

```
$!
$! Logicals for System Data Files
$!
$ DSE:=DEFINE/SYSTEM/EXECUTIVE
$!
$ DSE NETPROXY SYS$SPECIFIC:[SYSEXE]NETPROXY.DAT
$ DSE RIGHTSLIST SYS$SPECIFIC:[SYSEXE]RIGHTSLIST.DAT
$ DSE SYSUAF SYS$SPECIFIC:[SYSEXE]SYSUAF.DAT
$ DSE SYSUAFALT SYS$SPECIFIC:[SYSEXE]SYSUAFALT.DAT          ! Not used here
$ DSE SYSALF SYS$SPECIFIC:[SYSEXE]SYSALF.DAT                ! Not used here
$ DSE VMS$PASSWORD_HISTORY SYS$SPECIFIC:[SYSEXE]VMS$PASSWORD_HISTORY.DATA
$ DSE VMS$PASSWORD_DICTIONARY SYS$SPECIFIC:[SYSLIB]VMS$PASSWORD_DICTIONARY.DATA
$ DSE VMSMAIL_PROFILE SYS$SPECIFIC:[SYSEXE]VMSMAIL_PROFILE
```

With the exception of the queue database files, you can specify the location of the other system data files listed in Table 5.9 by defining similar logical names. Coordinating queue database files requires additional steps that are discussed in chapter 8.

Although these logical names are required only for VMScluster environments, you should consider defining them even on stand-alone systems. This facilitates conversion to a clustered environment should that ever be necessary.

The following system files cannot be shared among cluster members:

```
SYS$SYSTEM:VAXVMSSYS.PAR           ! System Parameters
SYS$MANAGER:OPERATOR.LOG           ! Operator Log File
SYS$MANAGER:ACCOUNTNG.DAT          ! Resource Accounting File
SYS$SYSTEM:NETNODE_LOCAL.DAT       ! Local DECnet Configuration
SYS$MANAGER:SECURITY.AUDIT$JOURNAL ! Security Audit Events
SYS$ERRORLOG:ERRORLOG.DAT          ! System Error Log File
SYS$SYSTEM:PAGEFILE.SYS            ! Page Files
SYS$SYSTEM:SWAPFILE.SYS            ! Swap Files
```

5.7 Miscellaneous Startup Issues

5.7.1 Dependency Issues

When you establish the order of startup procedures, remember the following startup dependencies:

- Disks must be mounted before the applications stored on them are started
- Terminal ports must be configured before printers are configured to them
- DECnet must be the first network product started (e.g., start DECnet before Local Area Transport [LAT])
- LAT must be started and configured before LAT print queues can be configured

5.7.2 Handling System Startup Errors

By default, a DCL command procedure aborts when an error or severe error is encountered. This behavior may be appropriate for some DCL command procedures, but not for startup procedures. Many errors are not severe enough to warrant terminating startup procedures. For example, if you have a hardware problem with terminal port TXA0:, and your startup procedures issue a $SET TERMINAL TXA0:, a severe error is generated, and the startup procedure aborts.

You can eliminate this problem by disabling the default DCL error handler with the following command inserted at the beginning of each site-specific startup procedure:

```
$ SET NOON                    ! Set No On Error
```

When the error handler is disabled, OpenVMS continues processing the remaining commands in each startup procedure, even after errors are encountered. Make certain that this behavior is acceptable for your particular startup procedures.

5.8 Optimizing System Startup

The following sections discuss several techniques for reducing system startup time, including:

- Console logging
- Using startup batch jobs
- Snapshot facility

5.8.1 Console Logging

Throughout the startup process, status, informational, and error messages are displayed on the console terminal. Even if you have a hard-copy console, a vast number of startup messages can easily span several printed pages. With a video terminal, startup messages quickly disappear from the screen. In either case, you can easily miss important information and critical error messages.

Even if you do notice errors during startup, it is difficult to pinpoint exactly where in the startup process the error occurred. You can facilitate troubleshooting by including the DCL SET VERIFY command in all your startup procedures. However, this causes every line of every procedure to be displayed on the console terminal during startup. Even if you have a video console terminal, your total system startup time will dramatically increase while you wait for output to be echoed to the terminal.

You can better detect startup errors and greatly reduce startup time by redirecting system startup output to a file. Under OpenVMS 5.0 you do this by adding the following DCL commands to each site-specific startup procedure. For example:

```
$! SYSTARTUP_Vxx.COM
$!
$ DEFINE SYS$OUTPUT SYS$MANAGER:SYSTARTUP.LOG
$ DEFINE SYS$ERROR SYS$MANAGER:SYSTARTUP.LOG
$ SET PREFIX "<!5%T>"          ! Enable time-stamps on log file
$ SET VERIFY
$ OPEN/WRITE CONSOLE OPA0:
$ SAY:=CALL DO_OUTPUT          ! Send output to log file and console
$ SAY "Starting SYSTARTUP_Vxx"
.
. rest of SYSTARTUP_Vxx.COM
.
$ SUBMIT/NOPRINT SYS$MANAGER:STARTUP_CHECK ! Check for startup errors
$! Cleanup and exit
$!
$ SAY "Completed SYSTARTUP_Vxx"
$ CLOSE CONSOLE
$ SET NOPREFIX
$ EXIT
$!
$DO_OUTPUT: SUBROUTINE
$ string = f$fao("!/!AS!AS!AS!/","*** ",p1," ***")
```

```
$ write console string
$ EXIT
```

The foregoing commands force all output to the file SYSTARTUP.LOG. This allows the startup procedure to complete quickly because it does not have to wait for startup messages to be echoed on the console terminal—even when SET VERIFY is enabled. In this example, near the end of SYSTARTUP, the procedure STARTUP_CHECK is submitted as a batch job. STARTUP_CHECK uses the OpenVMS Search Utility to scan SYSTARTUP.LOG for errors. If errors are found, STARTUP_CHECK sends an OpenVMS mail message to all users listed in the DIS_SYSMGR mail distribution list. See chapter 16 for more information on mail distribution lists.

An example STARTUP_CHECK is as follows:

```
$! STARTUP_CHECK.COM
$!
$ SET DEFAULT SYS$MANAGER
$ SEARCH/MATCH=OR/EXACT/WINDOW=6/OUTPUT=STARTUP_CHECK.TMP -
   SYPAGSWPFILES.LOG, SYCONFIG.LOG, SYLOGICALS.LOG, -
   SYSTARTUP.LOG, STARTUP_*.LOG -
   "-W-","-E-","-F-"           ! Warning, Severe Error, Fatal Error
$ MAIL/SUBJECT="Startup error check" -
   SYS$MANAGER:STARTUP_CHECK.TMP DIS_SYSMGR
```

The /WINDOW=6 qualifier forces the search utility to display the six lines that surround any matches.

[new V6.0]

Starting with OpenVMS 6.0, redirecting console output during system startup is simplified with the *console logging facility*. Instead of having to add commands to each startup procedure, you can redirect output to a file by issuing the following commands:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> STARTUP SET OPTIONS/OUTPUT=FILE/CHECKPOINTING
```

The /OUTPUT=FILE qualifier causes startup output to be written to the file SYS$SYSTEM:STARTUP.LOG rather than the console terminal. You can optionally add the /CHECKPOINTING qualifier to force time stamps to be written to the log file. Finally, modify STARTUP_CHECK to scan STARTUP.LOG for error messages.

5.8.2 Using Startup Batch Jobs

If there are many components in your startup procedures that can be executed independently of each other, you may be able to speed system startup by submitting the procedures as multiple batch jobs that execute simultaneously. This technique yields the most pronounced results when used on Digital's more powerful processors, especially systems with multiple processors.

For example, using the startup procedures discussed earlier in this chapter, assume that STARTUP_PRODUCTS and STARTUP_APPLICATIONS do not depend on each other. Instead of directly executing these command procedures, modify SYSTARTUP_Vxx.COM to submit them as batch jobs

```
$!
$! SYSTARTUP_VMS.COM
$!
$ ...
$ ...
$ submit/noprint/notify SYS$MANAGER:STARTUP_PRODUCTS
$ submit/noprint/notify SYS$MANAGER:STARTUP_APPLICATIONS
$ synchronize STARTUP_PRODUCTS          ! Wait for both to complete
$ synchronize STARTUP_APPLICATIONS
```

The SYCHRONIZE command is used to force the execution of SYSTARTUP to pause until both STARTUP_PRODUCTS and STARTUP_APPLICATIONS are complete. This ensures that all system startup procedures are complete before users are allowed to log in. If you use this technique, you should also modify STARTUP_CHECK to scan the log files that the batch jobs create (e.g., SYS$MANAGER:STARTUP_PRODUCTS.LOG).

5.8.3 Snapshot Booting

[new V6.0]

Beginning with OpenVMS 6.0 you can reduce startup time on stand-alone systems by using the Snapshot facility. With the Snapshot facility you can save the state of the system as it exists soon after a normal system boot to a file known as a *snapshot image*. The next time you need to reboot, you can boot from the snapshot image and eliminate the need to run most system startup procedures.

Currently, you can use the Snapshot facility only on stand-alone systems. However, when allowable, snapshot images can reduce boot time by 50%–

80%. For example, on a VAXstation 2000 boot time was reduced from 6.5 to 1.5 minutes.

To create a snapshot image, perform the following steps:

```
$ SET TERM/NOBROADCAST/PERM

$ @SYS$MANAGER:SNAPSHOT

This procedure shuts down the system and writes out memory to a system image file.


Do you wish to continue (Y/N): Y

Please ignore any "%SYSTEM-W-POOLEXPF, Pool Expansion Failure" messages.

%SNAPSHOT-I-BOOTSTYLE, Setting the BOOT_STYLE sysgen parameter to 1

%SNAPSHOT-S-DETACHED, System Snapshot Version of 24-OCT-1995 16:34:41.05

  Snapshot to run as a detached process in 60 seconds.

SYSTEM logged out at 23-FEB-1995 12:06:37.54
```

The Snapshot process sets a special system parameter (BOOT_STYLE) to indicate that the system should boot from the snapshot image rather than perform a normal full boot. As a result, the next time you boot the system, the following occurs:

```
83 BOOT SYS

-DUA1

%SNAPSHOT-I-OPEN, opening the snapshot file
%SNAPSHOT-I-VALIDATE, validating the snapshot file
%SNAPSHOT-I-READMAP, reading the snapshot file bitmaps
%SNAPSHOT-I-READ, reading the snapshot file
%SNAPSHOT-I-BUFFER, reading the I/O vector buffer
%SNAPSHOT-I-OVERWRITE, overwriting the boot structures
%SNAPSHOT-I-RESTART, restart to follow

Forest Facility Node BIRCH

Username: SYSTEM
Password:
```

```
Welcome to VAX/VMS Version T6.0-5KE

  Last interactive login on Tuesday, 23-FEB-1995 12:02
  Last noninteractive login on Tuesday, 9-FEB-1995 12:25

     You have 21 new Mail messages.

$ SHOW SYSTEM
  ...
```

You must perform a number of detailed steps to prepare for taking a snapshot. These steps that are discussed in detail in the *OpenVMS System Manager's Manual: Essentials.*

5.9 Conclusion

Startup procedures for larger systems can be as complex as a small software application. So you should apply standard application change management strategies to startup procedures. This includes developing consistent documentation standards and release procedures. For example, you may want to centralize development of all startup procedures on a single development system. Then you make all changes on the development system, test them, and distribute them to one or more production systems. Using this technique, you can deploy changes to multiple systems easily with a simple DCL copy command.

For more complex environments, you might consider using a formal code management system such as Digital's Code Management System (CMS). CMS also allows you to retrieve all prior versions of a piece of code no matter how many times it has been changed. Also, every time you make a change, CMS requests comments that are automatically added to a historical journal.

5.10 Additional References

Mehrewn, David, *VAX Systems SIG Session Notes*, "I Want it Back Now! How to Get Your VAX/VMS Up and Running FAST!" Digital (Fall 1990).

VMS Internals and Data Structures, chapter 31, "Operating System Initialization and Shutdown," Digital, 1991.

[OpenVMS 5.0]

VMS SYSMAN Utility Manual, Digital, June 1990.

VMS Systems Manager's Manual, chapter 2, "Guide to Setting Up a VMS System," Digital, April 1988.

[Open VMS 6.0]

OpenVMS System Management Utilities Reference Manual, part 17, "SYSMAN," Digital, March 1994.

OpenVMS System Manager's Manual, chapter 5, "Customizing the Operating System," Digital, March 1994.

VMScluster Systems for OpenVMS, chapter 4, "Preparing the Cluster, Operating Environment," Digital, March 1994.

6 SYSTEM SHUTDOWN

6.1 Chapter Objectives

- Minimize accidental system shutdowns
- Ensure continuity of VMSclusters when individual members are shut down
- Achieve data integrity by ensuring that application-specific shutdown procedures are performed
- Explore the shutdown process in detail
- Utilize DCL procedures to automate common system shutdown methods

6.2 Introduction

The following are typical situations that require a system shutdown:

- After a software installation or upgrade
- After changing nondynamic system parameters
- Before a power shutdown
- Before performing special booting procedures (e.g., stand-alone backup)

There are two basic shutdown techniques:

- Orderly shutdown
- Emergency shutdown

When you request an orderly shutdown, installed images are removed and all disks are dismounted. These steps ensure that any critical operations in progress are allowed to complete before the system is actually halted. This reduces the risk of data file and disk corruption.

When you perform an emergency shutdown, the system is halted in a more abrupt manner. An emergency shutdown is still better then simply turning the system off, but because data can be lost, you should use emergency procedures only when orderly shutdown procedures fail, or if you need to force a system crash dump (discussed later in this chapter).

6.3 Orderly Shutdown Issues

The Digital-supplied DCL command procedure SYS$SYSTEM:SHUTDOWN.COM is the primary mechanism for performing an orderly shutdown. Beginning with OpenVMS Version 6.0, you can also use the SYSMAN SHUTDOWN command, as follows:

```
$ @SYS$SYSTEM:SHUTDOWN
```

or

```
[New V6.0]
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> SHUTDOWN NODE
```

The following privileges are required to perform a shutdown:

```
AUDIT, CMKRNL, EXQUOTA, LOG_IO, NETMBX, OPER, SECURITY,
SYSNAM, SYSPRV, TMPMBX, WORLD
```

If you want to prevent users or operators from initiating an orderly shutdown, you should ensure that they are not authorized for one or more of the above privileges.

SYSMAN SHUTDOWN provides the same functionality as SHUTDOWN.COM except shutdown options are selected using command qualifiers rather than through an interactive scripting process. The following sections discuss the procedures for using SHUTDOWN.COM. SYSMAN SHUTDOWN is discussed later in this chapter.

After you invoke SHUTDOWN.COM you are prompted to select and configure the following shutdown options:

- Shutdown notification messages
- Disk spin down
- Site-specific shutdown procedures

- Automatic system reboot
- Other shutdown options

6.3.1 Shutdown Notification Messages

SHUTDOWN allows you to broadcast system messages to all users reminding them that the system soon will be shut down. Your responses to three SHUTDOWN questions determine the length of the notification period, advertised reboot time, and reason for the shutdown. For example:

```
How many minutes until final shutdown [0]: 10
When will the system be rebooted [later]: at 4:00 p.m.
Reason for shutdown [Standalone]: Performing Maintenance
```

The above responses would force a 10-minute notification period and the following message to be sent:

```
SHUTDOWN message on BIRCH from user SYSTEM at _BIRCH$OPA0: 14:10:07
BIRCH will shut down in 10 minutes; back up at 4:00 p.m.
Please log off node BIRCH.
Performing Maintenance
```

The shutdown message is rebroadcast throughout the notification period at decreasing time intervals. Once the notification period is over, the system shutdown begins. If you select a 0-minute notification period (the default), SHUTDOWN sends only a single notification message and immediately begins a system shutdown. Unless you know that no one is using the system, you should always select at least a 5- or 10-minute notification period. Try to be as descriptive and accurate as you can in choosing text for the reboot time and shutdown reason. This way, users will know what's going on and know when they can reasonably expect the system to become available.

You can enforce a minimum shutdown notification period by assigning a value to the system-wide logical name SHUTDOWN$MINIMUM_MINUTES. For example, to enforce a 10-minute minimum, issue the following command:

```
$ DEFINE/SYSTEM/EXECUTIVE SHUTDOWN$MINIMUM_MINUTES 10
```

If system shutdowns are usually performed by operations staff, you should establish a minimum shutdown notification period. To make your choice

permanent, add the logical name definition to your system startup procedures (see chapter 5).

By default, shutdown notification messages are sent to users cluster-wide even if only a single system is being shut down. This can confuse users when they receive notification messages that really don't affect them. You can restrict shutdown notification messages to affected nodes by defining the logical name SHUTDOWN$INFORM_NODES for each cluster member, as follows:

```
$ DEFINE/SYSTEM SHUTDOWN$INFORM_NODES "node" ! Where node = local node name
```

Again, to make this choice permanent, add the definition to system startup procedures (see chapter 5).

6.3.2 Spin-down Disks

You can request that all disk drives—excluding the system disk—be spun down as part of the shutdown process by responding YES to the following SHUTDOWN question:

```
Do you want to spin down the disk volumes [NO]:
```

Normally, you only need to select this option when you want to power-off the entire system and its peripherals. Not all disk drives support this option.

6.3.3 Site-specific Shutdown Procedures

If you respond YES to the following question, SHUTDOWN executes the site-specific shutdown procedure, SYS$MANAGER:SYSHUTDWN.COM, before stopping user processes:

```
Do you want to invoke the site-specific shutdown [YES]?
```

By default, OpenVMS simply deletes all active processes as part of the normal shutdown process. If you have applications that continuously update data files, such an abrupt action can cause file corruption. To prevent this, many applications and layered products have associated shutdown procedures that perform an orderly *application shutdown*. You should carefully review documentation for all Digital, third-party, and in-house applications and include appropriate shutdown commands in SYSHUTDWN. For example

```
$! SYSHUTDWN.COM
$!
$ @SYS$MANAGER:VPA$SHUTDWN          ! VAX Performance Advisor
$ @SYS$MANAGER:DB_SHUTDOWN          ! Database Application
$ @SYS$MANAGER:ACCT_SHUT            ! Financial Management System
```

If you don't have application-specific shutdown procedures, create an empty SYSHUTDWN.COM. This way, you can let it be invoked (the default shutdown option), but because it is empty, no commands are executed.

In VMScluster environments, shutdown procedures may have to be tailored further when all systems don't run the same applications. For example, if only cluster members BIRCH and LOTUS run VPA, SYSHUTDWN might look like this:

```
$! SYSHUTDWN.COM
$!
$ NODE = F$GETSYI("SCSNODE")
$ IF (NODE .EQS. "LOTUS") .OR. (NODE .EQS. "BIRCH") -
    @SYS$MANAGER:VPA$SHUTDWN        !VAX Performance Advisor
$ @SYS$MANAGER:DB_SHUTDOWN          !Database Application
$ @SYS$MANAGER:ACCT_SHUT            !Financial Management System
```

In complex cluster environments, it might be easier to establish a separate shutdown procedure for each cluster member. For example:

```
$! SYSHUTDWN.COM - Cluster Common Shutdown
$!
$ NODE = F$GETSYI("SCSNODE")
$ @SYS$MANAGER:SYSHUTDOWN_'NODE'    !Execute node-specific shutdown
$!
$! Common Shutdown Procedures
$!
$ @SYS$MANAGER:DB_SHUTDOWN          !Database Application
$ @SYS$MANAGER:ACCT_SHUT            !Financial Management System
```

6.3.4 Automatic Reboot

If you want the system to reboot immediately after the shutdown, respond "YES" to the following shutdown question:

```
Should an automatic system reboot be performed [NO]?
```

If you do not request an automatic reboot, the following message will be displayed on completion of the shutdown process:

Table 6.1 Default Cluster Quorum Adjustment after SHUTDOWN

Event	*Votes Present*	*Expected Votes*	*Quorum*	*Cluster State*
Cluster booted	4	4	3	OK
Shutdown member #1	3	4	3	OK
Shutdown member #2	2	4	3	Hang

```
SYSTEM SHUTDOWN COMPLETE - USE CONSOLE TO HALT SYSTEM
```

You can then halt the system and issue manual boot commands (e.g., to boot stand-alone backup) as described in chapter 4.

6.3.5 SHUTDOWN Options

The following SHUTDOWN question allows you to select one or more additional shutdown options:

```
Shutdown options (enter as a comma-separated list):

REMOVE_NODE          (VMScluster only)
CLUSTER_SHUTDOWN     (VMScluster only)
REBOOT_CHECK
SAVE_FEEDBACK
DISABLE_AUTOSTART    (new V5.5)
NONE

Shutdown options [NONE]:
```

Each of these options is described in the following sections.

REMOVE_NODE

An important issue to consider when performing a system shutdown is cluster quorum continuity. By default, when a cluster member is shut down, its contributed votes are removed, but quorum is *not* automatically recalculated. Therefore, after you shut down one or more voting members, the number of contributed votes eventually falls below the calculated quorum value, causing loss of quorum and suspension of user activity—a *cluster hang*. Table 6.1 demonstrates cluster hang for a four-member cluster in which each member contributes one vote.

Table 6.2 Quorum Adjustment with REMOVE_NODE Option Selected

Action	*Votes Present*	*Expected Votes*	*Quorum*	*Cluster State*
Cluster booted	4	4	3	OK
Shutdown #1 (REMOVE_NODE)	4	4	3	OK
After Shutdown #1	3	4	3	OK
Shutdown #2 (REMOVE_NODE)	3	3	2	OK
After Shutdown #2	2	3	2	OK
Shutdown #3 (REMOVE_NODE)	2	2	2	OK
After Shutdown #3	1	2	2	Hung
Invoke Interrupt Priority C (IPC) on #4	1	1	1	OK

This behavior can be avoided by selecting the REMOVE_NODE option during system shutdown. REMOVE_NODE forces a SET CLUSTER/EXPECTED_VOTES command to be issued just before the member is shutdown. SET CLUSTER/EXPECTED VOTES forces expected votes to be set to the current number of available votes *and* forces quorum recalculation.

However, when the SET CLUSTER command is issued, the node being shut down is still contributing its VOTES. Therefore, the quorum recalculation does *not* take into consideration the loss of votes about to occur from the node being shutdown. For example, in Table 6.2, after shutdown of the first cluster member, EXPECTED_VOTES is still 4; therefore, quorum is still calculated as 3. Only when the second cluster member is shut down is the quorum reduced to 2.

Because of this delayed quorum recalculation, you cannot successfully perform a REMOVE_NODE when only 2 cluster members remain. In Table 6.2, when the third cluster member is being shutdown, EXPECTED_VOTES is adjusted to 2, but this leaves quorum at 2. As soon as the shutdown of the third node completes, only 1 remaining vote is present, and therefore the cluster hangs. At this point, the only way to resume the cluster is to manually force quorum recalculation using the IPC utility (see chapter 4).

CLUSTER_SHUTDOWN

The CLUSTER_SHUTDOWN option is presented only if the system being shutdown is a VMScluster member. When a cluster member is shutdown, a state-transition occurs, and the remaining members negotiate a new cluster configuration. This behavior can cause problems when shutting down all cluster members at the same time. By default, a state-transition occurs after

each member is shut down. Eventually, too few cluster members remain to constitute quorum, forcing the cluster to hang without ever completing the shutdown.

When you shut down an entire cluster, specify the CLUSTER_SHUTDOWN option on all members. This forces each member to shut down to a point at which they are still considered cluster members. When all nodes are shut down to this common point, the cluster connection manager forces all nodes to complete the shutdown process simultaneously.

REBOOT_CHECK

The REBOOT_CHECK option forces the shutdown procedure to perform a reboot consistency check by confirming that key system files exist. If they exist, the shutdown process is allowed to continue, otherwise the shutdown aborts. If you suspect that system files may have been manipulated, select this option. For example, if the file SYSGEN.EXE did not exist and you initiated a REBOOT_CHECK, you would get the following error messages during a system shutdown:

```
$ @SYS$SYSTEM:SHUTDOWN
.
.
.
Shutdown options [NONE]: REBOOT_CHECK
.
%SHUTDOWN-E-OPENIN, Error opening SYS$SYSTEM:SYSGEN.EXE as input
%SHUTDOWN-E-NOREBOOT, The system will be unable to reboot
%SHUTDOWN-E-NOREBOOT, Correct above errors before shutting down
```

SAVE_FEEDBACK

During system operation, performance feedback data is stored in main memory. By default, this information is discarded after a system shutdown. If you select SAVE_FEEDBACK during shutdown, feedback data is written to the following file:

```
SYS$SYSTEM:AGEN$FEEDBACK.DAT
```

See chapter 18 for details on using old feedback data.

DISABLE_AUTOSTART=*minutes*

The DISABLE_AUTOSTART option is valid only if you are using the new Queue Manager introduced with OpenVMS 5.5. The Queue Manager is de-

scribed in more detail in chapter 8. With the new Queue Manager, all AUTOSTART queues are closed when the DISABLE AUTOSTART/QUEUES command is issued, preventing additional jobs from being submitted. Active jobs in closed queues can continue processing until a STOP/QUEUES/ON_NODE command is executed. At that point, active jobs either abort or fail-over to designated fail-over queues.

By default, the DISABLE AUTOSTART/QUEUES command is executed as soon as the shutdown process begins, and the STOP/QUEUES/ON_NODE is executed just before final shutdown. The DISABLE_AUTOSTART option allows you to delay the execution of the DISABLE AUTOSTART/QUEUES command. For example, if you specified a 60-minute shutdown notification period, you might want to use the DISABLE_AUTOSTART=10 option. Then autostart queues would not be closed until 10 minutes before shutdown.

You can force a default value for DISABLE_AUTOSTART by defining a system logical name as follows:

```
$ DEFINE/SYSTEM/EXECUTIVE SHUTDOWN$DISABLE_AUTOSTART "10"
```

To be effective, the number of minutes specified for the shutdown notification period must be greater than the number of minutes specified with the DISABLE_AUTOSTART option.

6.4 Orderly Shutdown Procedures

Before you actually activate SHUTDOWN you may want to perform the following steps:

- Disable future logins
- Check for system activity
- Send a shutdown notification message

6.4.1 Disable Future Logins

Although the shutdown process automatically disables system logins, you may want to disable them earlier to prevent users from logging in before the shutdown. Remember, disabling logins has no effect on users already logged in. In addition, users who are authorized for the OPER privilege can continue to log in. Use the following command to disable logins on a single system, or use SYSMAN to perform the same command on multiple systems in a VMScluster:

```
$ SET LOGINS/INTERACTIVE=0
%SET-T-INTSET, login interactive limit=0, current interactive value=5
```

or

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> SET ENVIRONMENT/CLUSTER
SYSMAN> DO SET LOGINS/INTERACTIVE=0
```

6.4.2 Check for System Activity

No matter how much you advertise a shutdown, some users do not get the message or forget that it is scheduled. Before you start the shutdown, use the MONITOR and SHOW utilities to check for system activity.

First check for interactive and batch logins:

```
$ SHOW USERS/NODE          ! List on this system only

     VAX/VMS User Processes at 22-APR-1995 15:34:25.34
Total number of users = 5, number of processes = 15

Username        Node       Interactive    Subprocess      Batch
SIMPSONB        BIRCH            1
SIMPSONM        BIRCH            1
.
.
.
```

Then, check for network jobs as follows:

```
$ SHOW SYSTEM/NETWORK

VAX/VMS T6.0-5KE on node BIRCH 13-APR-1995 06:20:49.10

Pid       Process Name   State  Pri  I/O      CPU         Page flts  Ph.Mem
0000008E  EVL            HIB    6    57    0 00:00:01.88       344      52     N
00000091  FAL_8194       LEF    4   109    0 00:00:03.80       435     400     N
00000092  FAL_8196       PFW    4     2    0 00:00:00.29        89     122     N
```

Only the standard network jobs such as EVL and NETACP should be listed. If there are processes named FAL_xxxxxxx, network file transfers may be in progress and you should investigate further before initiating the shutdown.

To see if any existing processing is active, use MONITOR as follows:

```
$ MONITOR PROCESSES/TOPCPU

                    VAX/VMS Monitor Utility
                    TOP CPU TIME PROCESSES
                         on node BIRCH
                     13-APR-1995 06:24:14

                      0         25        50        75        100
                      + - - - - + - - - - + - - - - + - - - - - -+
000000A2 DB_UPDATE    58  **********************
                      |         |         |         |          |
0000008F SYSTEM        8  ***
                      |         |         |         |          |
00000021 SWAPPER       1
                      |         |         |         |          |
00000029 JOB_CONTROL
                      |         |         |         |          |
00000028 AUDIT_SERVER
                      |         |         |         |          |
                      |         |         |         |          |
                      |         |         |         |          |
                      + - - - - + - - - - + - - - - + - - - - - -+
```

If you find a particular process and you want to know exactly what it is doing, use the SHOW PROCESS command, as follows:

```
$ SHOW PROCESS/ID=process-id-number
$ SHOW PROCESS/ID=A2

                  Process DB_UPDATE                    06:24:32

State             COM           Working set                837

Cur/base priority 3/3           Virtual pages             4315

Current PC        00004B97      CPU time        000:00:00:27.70

Current PSL       03C00000      Direct I/O                 635

Current user SP   7FE423D0      Buffered I/O               112

PID               000000A2      Page faults                897
```

```
UIC                     [SYSTEM]        Event flags             E0000001
                                                                80000000

BIRCH$DUA1:[SYS0.SYSCOMMON.][SYSEXE]SEARCH.EXE
```

Again, if you see activity you don't expect, investigate before proceeding with the shutdown.

6.4.3 Send Shutdown Notification

Although shutdown procedures issue shutdown notification messages, you might want to send your own messages manually as follows:

```
$ REPLY/ALL/BELL/NODE=node "System shutting down now, please log-off.
Call extension 1234 for assistance."
```

If you are shutting down only one node in a cluster, use the /NODE qualifier to restrict the message to just that node.

6.4.4 Invoke Shutdown

Once you are satisfied it is safe to begin shutdown, invoke SHUTDOWN:

```
$ @SYS$SYSTEM:SHUTDOWN
```

After invoking SHUTDOWN, you are prompted to enter several shutdown options. Some prompts display a default response enclosed in square brackets ([ ]). To select the default value, press <RETURN> or key in your selection.

The following shows a typical shutdown session:

```
SHUTDOWN -- Perform an Orderly System Shutdown on node BIRCH

How many minutes until final shutdown [0]: 10
Reason for shutdown [Standalone]: Performing Maintenance
Do you want to spin down the disk volumes [NO]? <RETURN>
Do you want to invoke the site-specific shutdown procedure [YES]? <RETURN>
Should an automatic system reboot be performed [NO]? <RETURN>
When will the system be rebooted [later]: at 11:00 a.m.
Shutdown options (enter as a comma-separated list):

  REBOOT_CHECK       Check existence of basic system files
  SAVE_FEEDBACK      Save AUTOGEN feedback information from this boot
  DISABLE_AUTOSTART       Disable autostart queues
```

```
Shutdown options [NONE]: DISABLE_AUTOSTART=5

SHUTDOWN message on BIRCH from user SYSTEM at _BIRCH$TTA3: 06:39:53
BIRCH will shut down in 10 minutes; back up at 11:00 a.m..
Performing Maintenance

%SHUTDOWN-I-DISAUTOSTART, the autostart queue feature will now be disabled with
          5 minutes remaining until shutdown

%SHUTDOWN-I-DISLOGINS, interactive logins will now be disabled

%SET-I-INTSET, login interactive limit = 0,current interactive value = 1

%SHUTDOWN-I-SHUTNET, the DECnet network will now be shut down

SHUTDOWN message...BIRCH will shut down in 5 minutes; back up at 11:00 a.m.

SHUTDOWN message...BIRCH will shut down in 3 minutes; back up at 11:00 a.m.

SHUTDOWN message...BIRCH will shut down in 2 minutes; back up at 11:00 a.m.

%SHUTDOWN-I-STOPQUEUES, the queues on this node will now be stopped

SHUTDOWN message...BIRCH will shut down in 1 minute; back up at 11:00 a.m.

SHUTDOWN message...BIRCH will shut down in 0 minutes; back up at 11:00 a.m.

%SHUTDOWN-I-SITESHUT, the site-specific shutdown procedure will now be invoked

Executing site-specific SHUTDOWN procedures

%SHUTDOWN-I-STOPUSER, all user processes will now be stopped

%SHUTDOWN-I-STOPAUDIT, the security auditing subsystem will now be shut down

%SHUTDOWN-I-REMOVE, all installed images will now be removed

%SHUTDOWN-I-DISMOUNT, all volumes will now be dismounted

%DISM-I-MARKEDDMT, _BIRCH$DUA0: has been marked for dismount

%%%%%%%%%%% OPCOM 13-APR-1995 06:50:47.04 %%%%%%%%%%%
Logfile was closed by operator _BIRCH$TTA3:

Logfile was BIRCH::DISK$USER1:[SYSFILES]OPERATOR_BIRCH.LOG;27

  SYSTEM SHUTDOWN COMPLETE - USE CONSOLE TO HALT SYSTEM
```

Table 6.3 SYSMAN SHUTDOWN Qualifiers

Qualifier	*SHUTDOWN.COM Question*	*Parameter*
/MINUTES_TO_SHUTDOWN =minutes	How many minutes until final shutdown:	P1
/REASON="Text"	Reason for shutdown:	P2
/REBOOT_TIME="Text"	When will the system be rebooted:	P5
/[NO]SPIN_DOWN_DISKS	Do you want to spin down the disk volumes:	P3
/[NO]AUTOMATIC_REBOOT	Should an automatic system reboot be performed?	P6
/INVOKE_SHUTDOWN	Do you want to invoke the site-specific shutdown?	P4
/REBOOT_CHECK	Option: REBOOT_CHECK	P7
/CLUSTER_SHUTDOWN	Option: CLUSTER_SHUTDOWN	P7
/REMOVE_NODE	Option: REMOVE_NODE	P7
/SAVE_FEEDBACK	Option: SAVE_FEEDBACK	P7
/DISABLE_AUTOSTART =minutes	Option: DISABLE_AUTOSTART=minutes	P7

6.4.5 SYSMAN SHUTDOWN

[New V6.0]

Beginning with OpenVMS 6.0, you can perform a system shutdown using the SYSMAN SHUTDOWN NODE command. SYSMAN still invokes SYS$SYSTEM:SHUTDOWN to perform the actual shutdown, but qualifiers have been added so you can initiate a system shutdown with a single command line, rather than having to follow an interactive script. You can also use the SYSMAN SET ENVIRONMENT command to select and shut down multiple systems or an entire cluster with a single command.

The syntax of the SYSMAN SHUTDOWN command is as follows:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> SHUTDOWN NODE /qualifiers
```

Table 6.3 illustrates valid SHUTDOWN qualifiers, the corresponding SHUTDOWN.COM question, and the parameter number.

Another advantage of SYSMAN SHUTDOWN is that by default it executes in a spawned subprocess, leaving you free to monitor the system while you wait for the shutdown to complete. Therefore, make sure you don't log off

after starting a shutdown; otherwise your subprocess will be deleted and the pending shutdown will be aborted. You can prevent this behavior by forcing the shutdown to be executed as an independent, detached process rather than as a spawned process. This occurs whenever you use the SET ENVIRONMENT command to select one or more cluster members. If you want to shut down only the local system, use the following command:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> SET ENVIRONMENT/NODE=local_node
SYSMAN> SHUTDOWN NODE /qualifiers
```

6.5 Emergency Shutdown Procedures

If normal shutdown procedures fail, or if they cannot be invoked, you can use the following emergency shutdown procedures.

6.5.1 OPCCRASH

During the last step of an orderly shutdown SYS$SYSTEM:OPCCRASH.EXE is executed, which dismounts the system disk and halts the system (or initiates a reboot if requested).

If an orderly shutdown fails, you may be able to force a shutdown by manually invoking OPCCRASH as follows:

```
$ RUN SYS$SYSTEM:OPCCRASH
```

OPCCRASH also examines the logical name OPC$NODUMP. If OPC$NODUMP is clear (the default), a crash dump file is created. During an orderly shutdown, OPC$NODUMP is set, preventing a crash dump every time you perform a shutdown.

6.5.2 Performing a Crash Dump

If you can't even invoke OPCCRASH (e.g., the system is completely hung), you can manually force a system crash dump by using console commands. The following summarizes the procedures for most Digital systems.

VAX Systems

```
!Set terminal control to LOCAL/ENABLE
^P, <BREAK>, or HALT button    ! Key sequence to switch to console mode
```

```
>>> @CRASH                          ! On processors with boot command files
```

or

```
>>> D/G F FFFFFFFF                  ! Put invalid address in Program Counter
>>> DEPOSIT PSL 041F0000            ! Raise IPL 31 on Interrupt Stack,
                                      (>>> D P 041F0000) on some systems
>>> CONTINUE                        ! Jump to invalid address
                                    ! at high IPL, thus forcing
                                    ! invalid exception on interrupt
                                    ! stack bug-check, and
                                    ! subsequent crash dump
```

AXP Systems

```
                                    ! Set terminal control to LOCAL/ENABLE
^P, <BREAK>, or HALT button         ! Key sequence to switch to console mode
>>> CRASH                          ! AXP 4000, 7000, 10000
```

or

```
>>> E -N 1E R0
>>> E PS
>>> E PC
>>> D PC FFFFFFFF00000000           ! Put invalid address in Program Counter
>>> D PS 1F00                       ! Raise IPL 31
>>> C                               ! Jump to invalid address
                                    ! at high IPL, thus forcing
                                    ! invalid exception on interrupt
                                    ! stack bug-check, and
                                    ! subsequent crash dump
```

6.5.3 Sizing the System Dump File

When a crash dump occurs, the system attempts to copy the entire contents of physical memory and error log buffers to the system dump file (SYS$SYSTEM:SYSDUMP.DMP). *The crash dump file must be located on the system disk.* To ensure a complete crash dump, you should size your crash dump file using the following formulas:

VAX Systems

```
Dump-file-size
= physical-memory (pages) + (ERRORLOGBUFFERS * ERLBUFFERPAGES) + 1
```

AXP Systems

```
Dump-file-size
= physical-memory (pagelets) + (ERRORLOGBUFFERS * ERLBUFFERPAGES) + 2
```

You can determine the number of pages of physical memory using the SHOW MEMORY command. On AXP systems, SHOW MEMORY displays the number of AXP pages. To calculate the number of pagelets, multiply the number of pages by the number of blocks per AXP page, as follows:

```
$ BLKS_PER_PG = F$GETSYI("PAGE_SIZE")/512
$ PAGELETS = BLKS_PER_PG * pages    ! Get "pages" from SHOW MEMORY
```

You can determine the number of error log buffers and their corresponding size by examining the system parameters ERRORLOGBUFFERS and ERLBUFFERPAGES. Once you have calculated the required dump file size, create it as follows:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> CREATE SYS$SYSTEM:SYSDUMP.DMP/SIZE=size
```

If your system has hundreds of megabytes of physical memory, you may find that your system disk does not have enough space to support a full crash dump. In this case, set the system parameter DUMPSTYLE to 1 to configure the system for selective crash dumps. When the system performs a selective crash dump, it copies only the most critical parts of memory, thus requiring a smaller crash dump file. To force AUTOGEN to calculate the proper dump file size, add the following lines to MODPARAMS.DAT and execute AUTOGEN:

```
$ EDIT MODPARAMS.DAT
! MODPARAMS.DAT
...
!DUMPFILE=0              ! Remove any references to DUMPFILE
DUMPSTYLE=1

[exit and save changes]

$ @SYS$UPDATE:AUTOGEN SAVPARAMS TESTFILES
```

AUTOGEN sizes the dump file to be approximately equal to the number of pages allocated to the system plus a fraction of pages allocated to processes (10% to 50%). This usually results in a dump file that is 40% to 70% of

physical memory. As an alternative, you can use these guidelines and manually size your dump file using SYSGEN.

Although a selective dump requires less disk space, frequently a selective dump does not contain enough information to determine the cause of the crash.

6.5.4 Using the Page File as the System Dump File

To conserve system disk space you can optionally use the primary page file as a dump file. If there is no crash dump file, OpenVMS copies dump information to the primary page file (SYS$SYSTEM:PAGEFILE.SYS). Normally, the page file is initialized after each system boot, potentially invalidating an earlier crash dump. To use the page file as a dump file, you have to prevent this behavior by setting the system parameter SAVEDUMP to 1. This preserves crash dump pages written to the page file until they are manually copied or released.

Because much of the page file is unusable after a crash dump, you should make the page file large enough to hold a dump *and* add several thousand pages to allow the system to boot. Once a system boots after a crash dump, you can release allocated crash dump pages, as follows:

```
$ ANALYZE/CRASH SYS$SYSTEM:PAGEFILE.SYS
SDA> COPY $DISK2:[TEMP]SYSDUMP.SAV    ! Copy to alternate disk for later analysis
```

If you have no place to copy the dump, perform analysis and release pages as follows:

```
$ ANALYZE/CRASH/RELEASE SYS$SYSTEM:PAGEFILE.SYS
```

6.5.5 Establishing a Common Dump File

Normally, each system requires a dedicated system dump file. Even in a modestly sized VMScluster, it is usually impossible to allocate enough system disk space to support a full crash dump for all cluster members. As an alternative, you can establish a shared common-dump file, as follows:

1. Rename one dump file to the common root. Make sure it is large enough to support a full dump of any single cluster member. *You must not name it SYSDUMP.DMP.*

   ```
   $ RENAME $1$DUA0:[SYS0.SYSEXE]SYSDUMP.DMP -
     SYS$COMMON:[SYSEXE]SYSDUMP-COMMON.DMP
   ```

2. Rename the dump files on the existing cluster members so that they will not be used.

```
$ RENAME $1$DUA0:[SYS1.SYSEXE]SYSDUMP.DMP SYSDUMP.SAV
$ RENAME $1$DUA0:[SYS2.SYSEXE]SYSDUMP.DMP SYSDUMP.SAV
```

3. Create an alias file entry in each system root that points to the common dump file.

```
$ SET  FILE/ENTER=$1$DUA0:[SYS0.SYSEXE]SYSDUMP.DMP - ! [SYS0] root
  SYS$COMMON:[SYSEXE]SYSDUMP-COMMON.DMP

$ SET  FILE/ENTER=$1$DUA0:[SYS1.SYSEXE]SYSDUMP.DMP - ! [SYS1] root
  SYS$COMMON:[SYSEXE]SYSDUMP-COMMON.DMP

$ SET  FILE/ENTER=$1$DUA0:[SYS2.SYSEXE]SYSDUMP.DMP - ! [SYS2] root
  SYS$COMMON:[SYSEXE]SYSDUMP-COMMON.DMP
```

4. Reboot the entire cluster.

5. Delete the old dump files.

```
$ DELETE  $1$DUA0:[SYS0.SYSEXE]SYSDUMP.SAV;
$ DELETE  $1$DUA0:[SYS1.SYSEXE]SYSDUMP.SAV;
$ DELETE  $1$DUA0:[SYS2.SYSEXE]SYSDUMP.SAV;
```

Do not delete the old dump files until *after* you perform a system reboot. You should never delete an active dump file. If a system crash occurs before the next reboot, the system overwrites the disk blocks previously allocated by the dump file—even if they now are used for some other purpose.

6.6 Automating Shutdown

Although shutdown procedures depend on system configuration (e.g., clustered as opposed to stand-alone), each system is usually shut down using the same procedures each time. Once you've determined the shutdown methods you commonly use, you can create automated procedures to streamline the process. The goal is not to make it easier to initiate a shutdown but to ensure that systems are consistently shut down using the correct procedures.

In the same way that SYSMAN SHUTDOWN passes parameters to SHUTDOWN.COM, you can create your own DCL command procedures that use the same technique. Parameters are passed to SHUTDOWN.COM using the standard DCL parameters P1 through P7 as indicated in Table 6.3.

Next, you can create a simple DCL menu system that allows you to select the most common shutdown types you perform. For example:

```
                        SHUTDOWN MENU

1 - RETURN      - Return to previous menu
2 - REBOOT      - Shutdown now with immediate reboot
3 - REBOOT5     - Shutdown in 5 minutes, with immediate reboot
4 - NORMAL      - Shutdown now without reboot
5 - NORMAL5     - Shutdown in 5 minutes, without reboot
6 - POWEROFF    - Shutdown, prepare to shutoff power (spin down disks)

Enter choice:
```

Within the DCL menu procedure, option 2 could execute the following command:

```
$ @SYS$SYSTEM:SHUTDOWN 0 REBOOT NO YES LATER YES NONE
```

or, beginning with OpenVMS 6.0:

```
$ SYSMAN:==$SYSMAN
$ SYSMAN SHUTDOWN NODE/MINUTES=0/REASON="REBOOT"-
   /NOSPIN_DOWN_DISKS/INVOKE_SHUTDOWN/REBOOT_TIME="LATER"-
   /AUTOMATIC_REBOOT
```

The DCL command procedure SHUTDOWN_MENU.COM provides an example of a shutdown menu with several preconfigured automated shutdown options. The source code for this command procedure is located on the diskette included with this book (see Appendix E).

6.7 Conclusion

Proper shutdown preparation is as essential as the shutdown procedures themselves. All shutdowns should be scheduled as far in advance as possible so that users are accommodated. Once a shutdown is scheduled, it should be communicated so that even infrequent users will receive notification. You don't want to schedule downtime weeks in advance only to have to reschedule because someone didn't know.

Users should be notified by OpenVMS Mail and login announcement messages. Chapter 16 ("Mail Management") discusses how to establish mail distribution lists that facilitate sending mail messages to many users. Chapter 5

("System Startup Procedures") describes the system logical name SYS$ANNOUNCE, which establishes a login announcement message.

Once you initiate a shutdown, you should give users 5 to 10 minutes notice with notification messages. These messages are sent as broadcasts to all users using the REPLY broadcast messages. However, users' terminals must be configured to receive these messages. If terminal ports have the /NOBROADCAST characteristic set, broadcast messages will not be delivered. You must set all terminal ports to /BROADCAST during system startup (see STARTUP_SETTERM.COM in chapter 5). Also, watch for applications that disable broadcast messages after system startup.

Many problems can occur during system shutdown or after a subsequent system reboot. You should always schedule adequate time to resolve problems. The most common unscheduled downtime occurs as a result of underestimating scheduled downtime requirements. A good rule of thumb is to schedule enough downtime to perform at least two or three complete boot cycles (on many systems this can be 30 to 45 minutes each).

6.8 Additional References

VAX/VMS Internal and Data Structures, section 31.5, "System Shutdown," Digital Press, 1991.
Guide to Setting up a VMS System, Digital, April 1988.
VAXcluster Manual, pp. 7–27, Digital, November 1991.

[OpenVMS 6.0]

OpenVMS System Manager's Manual, section 4.8.2, "Performing an Orderly Shutdown," Digital, March 1994.
OpenVMS System Management Utilities Reference Manual, part 17, "SYSMAN," March 1994.

7 LICENSE MANAGEMENT

7.1 Chapter Objectives

- Promote a better understanding of Digital's software licensing practices and policies
- Provide methods for selecting license types and optimizing their use
- Discuss Digital's new Personal Use and Concurrent Use license types
- Ensure system availability through proper management of software licenses

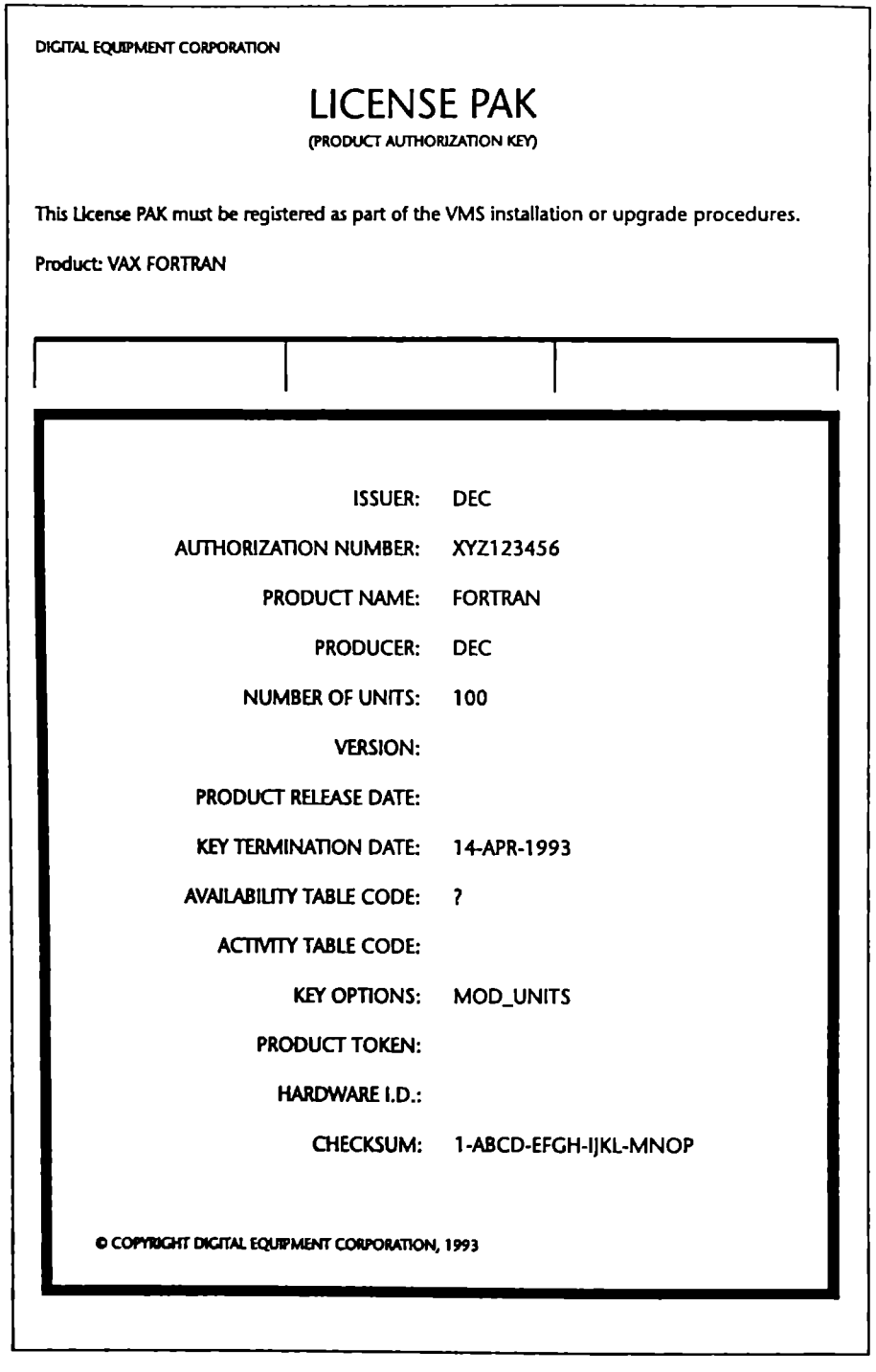
7.2 Introduction

As a system manager, you are responsible for installing, upgrading, and managing the software products on the systems you support. Consequently, you must ensure that these products are used in accordance with terms and conditions of associated software license agreements; otherwise, you or your organization could face severe civil and criminal penalties.

Before OpenVMS Version 5.0, Digital products were activated simply by installing the product with the VMSINSTAL utility. You could determine exactly which products you were properly licensed for only by reviewing original purchase orders and invoices. If you had multiple systems, this process could take weeks. Now, Digital distributes a paper Product Authorization Key (PAK) (Figure 7.1) whenever you purchase a software license. Before a product can be activated, the PAK for the product must be registered and loaded on the system or VMScluster it is designated for. Every time access to a product is attempted, the product checks to see if a corresponding PAK has been loaded. If it has, access is granted; otherwise, access is denied.

PAKs are required for the OpenVMS operating system and DECnet networking products. Users are not even able to log in to the system if these PAKs

Figure 7.1 Sample License Product Authorization Key



DIGITAL EQUIPMENT CORPORATION

LICENSE PAK

(PRODUCT AUTHORIZATION KEY)

This License PAK must be registered as part of the VMS installation or upgrade procedures.

Product: VAX FORTRAN

ISSUER:	DEC
AUTHORIZATION NUMBER:	XYZ123456
PRODUCT NAME:	FORTRAN
PRODUCER:	DEC
NUMBER OF UNITS:	100
VERSION:	
PRODUCT RELEASE DATE:	
KEY TERMINATION DATE:	14-APR-1993
AVAILABILITY TABLE CODE:	?
ACTIVITY TABLE CODE:	
KEY OPTIONS:	MOD_UNITS
PRODUCT TOKEN:	
HARDWARE I.D.:	
CHECKSUM:	1-ABCD-EFGH-IJKL-MNOP

© COPYRIGHT DIGITAL EQUIPMENT CORPORATION, 1993

are not registered and loaded correctly. Similarly, when you perform system configuration changes, you may need to update existing PAKs; otherwise, products can be accidentally deactivated. As you can see, system availability depends heavily on proper license management.

7.3 Digital Software Licenses

You do not purchase Digital software—you purchase a license that grants you the *right to use* software under specific terms and conditions. The following are some of the key points of Digital's software licensing terms and conditions. For more detailed information, refer to *A Practical Guide to VMS Software License Management and Tools.*

7.3.1 Licensed Version

Digital software licenses grant the right to use a specific version of a product (e.g., OpenVMS 5.0). If you want to use future versions (e.g., OpenVMS 5.1, 5.2, etc.) you must either purchase ongoing software support that includes the right to use new versions, or you must purchase an update license.

7.3.2 Product Categories

Digital licenses products in two general categories

- System integrated products
- Layered products

System Integrated Products

System integrated products (SIPs) are software products that are integrated into the operation of a hardware system. For example, OpenVMS, DECnet, and VMScluster software are SIPs because they are tightly integrated with processor hardware. Similarly, DECserver 200 software is a SIP because it is integrated with a DECserver 200 terminal server.

SIPs are generally licensed using only *traditional licenses.* Traditional licenses are priced on the basis of the processing power of the system and allow unlimited use of the licensed product on that system only.

Layered Products

Most other Digital products, such as C, Fortran, and Notes, are classified as layered products because they provide layered rather than integral function-

ality. Layered products are licensed using one or more of the major license types: traditional, ClusterWide, Personal Use, and Concurrent Use. Not all license types are available for every product.

7.3.3 Transferability

Digital software licenses are either *relicensable* or *non-relicensable*. SIPs are generally relicensable, and layered products are generally non-relicensable.

Relicensable products *can* be transferred outside your organization; however, they must be transferred with the hardware systems with which they were initially licensed. All transfers must be approved by Digital. You can apply for approval by submitting a relicense request.

Non-relicensable products *cannot* be moved outside your organization; however they *can* be transferred *within* your organization, as long the legal entity does not change. Licenses *can* be transferred from one system to another (although an upgrade fee may be required). You can transfer non-relicensable products without notifying Digital.

7.4 License Types

The following sections present the various licenses types offered by Digital and discuss the costs in a variety of situations. The prices listed in this book illustrate the general cost differences between license types—they do not represent current Digital prices. For recent pricing data, refer to Digital's *U.S. Software Price List.*

Digital products are primarily licensed using one of four license types.

Traditional

Traditional licenses allow unlimited use of a software product on a single processor. The cost of the license is somewhat proportional to the central processing unit (CPU) power of the system.

For example, the following is a listing for a VMScluster license:

Description	*Part #*	*Price*
VMScluster (for VAX 6410)	QL-VBRA2-AA	$12,261

Because they lack the flexibility of other license types, traditional licenses are usually less desirable; however, many Digital products are licensed only with traditional licenses.

ClusterWide

ClusterWide licenses are available for *either* a single stand-alone system or for a VMScluster that consists of several systems. Each ClusterWide license has an associated *license units* value (e.g., 100, 200, 300) that specifies the amount of use the license provides. Each processor has an associated *license unit requirement* that specifies the number of license units required for a product to be used on that system.

You can view the license unit requirements for a system by executing the following command:

```
$ SHOW LICENSE/UNIT_REQUIREMENT     ! use /CHARGE_TABLE for pre-V6.0

VMS/LMF Charge Information for node BIRCH
This is a VAXstation 2000, hardware model type 19
.
.
Type: D, Units Required: 100   (VMS Workstation)
Type: E, Units Required: 50    (System Integrated Products)
Type: F, Units Required: 10    (Layered Products)
```

For a stand-alone system to successfully load a ClusterWide license, the license unit value must be greater than or equal to the license unit requirements of the system. Once you load the license, the product can be used without limit on that system. For example, a VAX 6410 has a license unit requirement of 600 units. Thus, to successfully load and activate a Fortran license, the license must provide at least 600 license units. This license is listed as follows:

Description	*Part #*	*Price*
VAX Fortran (600 Units)	QL-100A9-JJ	$13,900

To use a product on several *nonclustered* systems, you must purchase a separate ClusterWide license for each system.

To use a product on several systems in the same VMScluster, purchase a license that has a license unit value equal to or greater than the *total* license unit requirements of all systems in the cluster. For example, to run Fortran on a cluster of three VAX 6410 systems, purchase a license that has 1800 (3 * 600) license units. A ClusterWide Fortran license, which provides 1800 license units, is listed as follows:

Description	*Part #*	*Price*
VAX Fortran (1800 Units)	QL-100A9-JM	$24,288

If the VAX 6410 systems were not a part of a VMScluster, you would have to purchase three separate 600 unit licenses. In this case, licensing Fortran for a cluster costs about $17,000 (3 * $13,900 – $24,288) less than licensing the same product on three stand-alone systems.

You do *not* have to license a product to run on all members of a VMScluster. In the foregoing example, to license two of the three VAX 6410 systems for Fortran, you need to purchase only one 1200 unit license. ClusterWide licenses are most appropriate when a large number of users need access to a product on one or more cluster members.

Concurrent Use

A Concurrent Use license allows a specific number of users to concurrently use a product on a single processor or any member of a VMScluster. For example, a four-user Concurrent Use Notes license allows up to four users simultaneous access to the Notes product. You might have a total of ten people who want to use the product, but only four can use it at the same time.

To compare costs, an *unlimited* Notes license for a VAX 6410 costs the following:

Description	*Part #*	*Price*
VAX Notes (600 units)	QL-960A9-JJ	$16,292

On the other hand, a four-user Concurrent Use license costs the following:

Description	*Part #*	*Price*
VAX Notes (1 concurrent)	QL-960A9A-3B	$1998 * 4 = $7992

In this case, Concurrent Use licensing costs $8000 less than unlimited ClusterWide licensing. However, if more than eight users need access, ClusterWide licenses would be less expensive.

Concurrent Use licenses are ideal when a large user group needs *infrequent* access to a product.

Personal Use

Personal Use licenses allow a specifically named user (based on OpenVMS username) unlimited access to a product. When assigned to a stand-alone system, a Personal Use license allows unlimited use by the named user on that system. When assigned to a VMScluster, a Personal Use license allows the named user unlimited access to the product from any member of the cluster.

For example, consider a cluster containing three VAX 6410 systems. To license Fortran for unlimited use on the entire cluster would cost more than $24,000. However, if only ten specific users needed access (e.g., John Smith, Jim Jones, etc), you could purchase ten Personal Use licenses as follows:

Description	*Part #*	*Price*
VAX Fortran 1 Personal Use	QL-100AA-2B	$1,444 * 10 = $14,440

In this case, the Personal Use licenses would cost $10,000 less than an unlimited ClusterWide license. However, if you had more than 16 specific users who needed to access Fortran, a ClusterWide license would be less expensive. In this example, if you later added another VMScluster member, the same Personal Use licenses would provide access to Fortran on that system as well. Another benefit is that if users move to another system, you can transfer the license with them.

Personal Use licenses are ideal when you have a specific set of users who need access to a product, especially if you need the flexibility to frequently move users from one system to another.

7.5 License Management Facility (LMF)

You manage digital licenses with the license management facility (LMF), which consists of LMF tools and the License Database.

7.5.1 License Database

The license database is an electronic representation of the paper PAKs that you receive from Digital when you purchase a software license. There are three components to the license database.

Permanent Database

By default, OpenVMS assumes that the permanent database is stored in the file

```
SYS$COMMON:[SYSEXE]LMF$LICENSE.LDB
```

You can optionally define a system logical name, LMF$LICENSE, to locate the permanent database in an alternative location as follows:

```
$ DEFINE/SYSTEM/EXEC LMF$LICENSE DISK$USER1:[SYSFILES]LMF$LICENSE.LDB
```

When you issue LICENSE commands, operations are performed on the default license database. To specify an alternative license database, add the /DATABASE qualifier to the LICENSE command.

If you maintain a common-environment VMScluster, you should define LMF$LICENSE so that all cluster members reference a common permanent database. If you maintain a multiple-environment VMScluster and cannot establish a shared license database, you can maintain multiple databases; however they should be exact duplicates of each other.

Volatile Database

Each system maintains an in-memory, volatile database that contains all the licenses that were successfully loaded either at system startup or manually with the LICENSE LOAD command. When you attempt to access a product, the product checks for a valid license in the volatile database—not in the permanent database.

Cluster-wide Database

Within a VMScluster, cluster-wide data structures are maintained that track license usage by cluster members. This is the primary reason why you must ensure that all cluster members load exactly the same licenses.

Consider a two-member, multiple-environment cluster (BIRCH and LOTUS) in which each system maintains its own LMF$LICENSE file. Assume that each system requires 100 units to run FORTRAN. Also assume that you register one 100 unit license in the license database of each system. When you attempt to load the license on BIRCH, LMF confirms that the number of units provided (100) meets the unit requirements of the system. LMF loads the license successfully and updates the cluster-wide database to reflect 100 units of use.

When you later attempt to load the FORTRAN license on LOTUS, LMF confirms that the license provides 100 units (enough to satisfy the unit requirements of the system); however, it also detects that the cluster-wide database shows that 100 units are already used by BIRCH, and the load attempt fails.

To avoid this type of problem, you must ensure that the LMF$LICENSE database files are identical for all cluster members. In this example, both 100 unit FORTRAN PAKs should be loaded in the license databases of both cluster members.

7.5.2 LMF Tools

You perform LMF functions with either the LICENSE and SHOW LICENSE DCL commands or with the VMSLICENSE.COM command procedure. The remaining examples in this chapter focus on using DCL commands. However, you can also perform most of the same functions by invoking the equivalent option from the VMSLICENSE menu.

LICENSE and SHOW LICENSE Commands

The LICENSE command has the following basic syntax:

```
$ LICENSE command product /qualifiers
```

For example,

```
$ LICENSE REGISTER VAX-VMS /....
```

Table 7.1 summarizes each of the LICENSE commands.

If your license database contains multiple licenses for the same product, you must add one or more of the following qualifiers to all LICENSE commands to uniquely specify the license:

```
/AUTHORIZATION=string
/ISSUER=string
/PRODUCER=string
```

In addition to the LICENSE commands, you can use the DCL command SHOW LICENSE to perform several additional license functions, as follows:

```
$ SHOW LICENSE ! Displays active licenses on the current node
```

To display the license unit requirement table (LURT) for your system, issue the following command:

```
$ SHOW LICENSE/UNIT_REQUIREMENT     ! Or /CHARGE_TABLE for pre-V6.0
```

You can list any active license that will terminate within 1000 days as follows:

```
$ SHOW LICENSE/WARNING_INTERVAL=1000     ! V6.0 Only
```

Table 7.1 LICENSE Commands

Command	*Function*
AMEND	Update a license with a Product Authorization Amendment (PAAM) Note: PAAMs are *not* currently supplied by Digital
CANCEL	Replaced by MODIFY/TERMINATION_DATE=date
COPY	Copy a license from one license database to another and disable the license in the source database
CREATE	Create a new (empty) license database in the directory specified by the /DATABASE qualifier
DELETE	Permanently delete a license from the license database
DISABLE	Prevent a license from loading at system startup time or manually with the LOAD command
ENABLE	Allow a license to load at system startup time or manually with the LOAD command
ISSUE	Create a PAK replica of a license
LIST	Display the contents of the license database
LOAD	Activate a license on the current node
MODIFY	Used to modify a license: include or exclude list, reservation list, termination date, or units
MOVE	Copies a license from one license database to another but deletes the license from the source database
REGISTER	Adds a license to the license database and enables it
START	Activates all registered and enabled licenses on the current node (subject to include and exclude lists). Automatically performed during site-independent startup procedures
UNLOAD	Deactivates a license on the current node

VMSLICENSE

The command procedure VMSLICENSE.COM allows you to perform the same functions as the DCL commands described above but from a simple menu system. To invoke VMSLICENSE, enter the following command:

```
$@SYS$UPDATE:VMSLICENSE

VMS License Management Utility Options:

  1. REGISTER a PAK
  2. AMEND a PAK
```

```
  3. CANCEL a PAK
  4. LIST a PAK
  5. MODIFY a PAK
  6. DISABLE a PAK
  7. DELETE a PAK
  8. COPY a PAK
  9. MOVE a PAK
 10. ENABLE a PAK
 11. SHOW the licenses loaded on this node
 12. SHOW the unit requirements for this node

 99. EXIT this procedure

Type '?' at any prompt for a description of the information
requested. Press Ctrl/Z at any prompt to exit this procedure.

Enter one of the above choices [1]
```

The function of each of the foregoing options is discussed in the previous section.

7.5.3 Installing a PAK

To make a license available to a system, you must perform the following steps:

- Register the paper PAK for the product into the permanent license database (LMF$LICENSE)
- Ensure that the registered license is enabled for loading
- Load the license into the volatile database of the system

The following commands demonstrate the process of installing a VAX-VMS license on node BIRCH:

```
$ LICENSE REGISTER VAX-VMS -          ! Register and Enable PAK
/ISSUER=DEC -
/AUTHORIZATION=ALL-WM-92098-2106 -
/PRODUCER=DEC -
/UNITS=0 -
/TERMINATION_DATE=14-APR-1995 -
/OPTIONS=(NO_SHARE) -
/CHECKSUM=1-AFKJ-EMCN-NEHP-WXYZ

$ LICENSE LOAD VAX-VMS                ! Load (activate) license
```

If you have many licenses, you should consider registering PAKs by creating DCL command procedures, rather than entering them interactively. For example, you could put the LICENSE REGISTER command in a file called REGISTER_VAX-VMS_BIRCH.COM. To register the PAK, execute the command procedure, as follows:

```
$ @REGISTER_VAX-VMS_BIRCH.COM
```

By creating command procedures, you can easily reregister PAKs in the event the license database becomes corrupted, and you can register new PAKs simply by copying a command procedure for a similar PAK and making a few edits.

7.5.4 Enabling and Disabling PAKs

PAKs registered in the permanent database are either *enabled* or *disabled*. All newly registered PAKs are automatically enabled. Enabled PAKS are automatically loaded into the volatile database during system startup. Disabled PAKs are not loaded.

To temporarily disable automatic loading of a PAK perform the following procedure:

```
Select DISABLE PAK or ENABLE PAK from the VMSLICENSE Main Menu
```

or

```
$ LICENSE DISABLE product-name
$ LICENSE ENABLE product-name
```

7.5.5 Loading a PAK

All enabled PAKs contained in the permanent database are automatically loaded into the volatile database of each system during system startup. You can manually load and unload PAKs after system startup using the following commands:

```
$ LICENSE LOAD product-name
$ LICENSE UNLOAD product-name
```

If you need to load or unload PAKs on several systems in a VMScluster, use the SYSMAN utility as follows:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> SET ENVIRONMENT/CLUSTER
SYSMAN> LICENSE LOAD product-name  ! Do it on all VAXcluster members
```

7.5.6 Deleting a PAK

If you receive a temporary PAK that you later replace with a new one, you may want to permanently remove the old PAK from the license database, as follows:

```
$ LICENSE DELETE product-name
```

Of course, you should ensure that you have a backup of the PAK, if for some reason you need to reregister it.

7.5.7 Restricting Product Access

In a VMScluster environment, you may need to restrict loading of PAKs to specific systems. By default, during system startup, OpenVMS attempts to load all PAKs contained in the permanent database. If the license units of a PAK are insufficient to meet the requirements of the entire cluster, some systems display the following error message during system startup:

```
%LICENSE-W-NOLOAD, license was not loaded for PRODUCT
-LICENSE-F-EXCEEDED, attempted usage exceeds active license limits
```

For example, if you have a VMScluster containing three VAX 6410 systems, a VAX Fortran PAK providing 1800 license units is required to support the entire cluster. If you have a PAK that provides only 1200 units, you will receive error messages when you boot the third system.

In this case, if it is important to provide Fortran access on two specific systems, specify an INCLUDE or EXCLUDE list for the PAK. For example, if you want Fortran activated only on nodes LOTUS and MANGO, use the following command:

```
$ LICENSE MODIFY/INCLUDE=(LOTUS, MANGO) FORTRAN
```

If you use Concurrent Use PAKs, you may have problems when unauthorized users access a product and prevent access by authorized users. To solve this problem, you can restrict access to the product to a specific set of users by establishing a RESERVE list as follows:

```
$ LICENSE MODIFY/RESERVE=(username1, username2, etc.) product-name
$ LICENSE MODIFY/RESERVE=(SIMPSONB, SIMPSONH) FORTRAN
```

If you use Personal Use licenses, you *must* establish a RESERVE list for the specific users who are licensed to use the product.

Once you have established a RESERVE list, you can add and remove names using the following commands:

```
$ LICENSE MODIFY/RESERVE=(SIMPSONB)/REMOVE FORTRAN      ! Remove a user
$ LICENSE MODIFY/RESERVE=(SIMPSONM)/ADD FORTRAN         ! Add a user
```

7.6 Additional References

A Practical Guide to VMS Software License Management and Tools, Digital (November 1991). Available from DECdirect (Order #EJ-31899-76).

[VMS 5.0]

U.S. Software Price List. Available from Digital Publishing and Circulation.
VMS License Management Utility Manual, Digital, November 1991.

[VMS 6.0]

OpenVMS License Management Utility Manual, Digital, March 1994.

8 QUEUE MANAGEMENT

8.1 Chapter Objectives

- Present methods for configuring and naming print and batch queues
- Focus on Centralized Queue Manager (introduced with OpenVMS Version 5.5)
- Provide guidelines for configuring queues and Queue Manager to ensure proper fail-over
- Discuss procedures for defining forms and device control libraries

8.2 Introduction

Because many changes have been made to the queue management subsystem over the last several years, it can be difficult to remember what features are available in each OpenVMS version. To clarify this, Table 8.1 summarizes queue management changes that have occurred between OpenVMS 5.4 and OpenVMS 6.0.

The first set of queue management changes was introduced in OpenVMS 5.5. At the same time, Digital provided a version A5.5 that was equivalent to version 5.5 *less* the queue management changes. This was to provide a smoother migration path for sites that needed version 5.5 functionality but required additional time to make changes to queue configurations. In terms of queue management, OpenVMS VAX 5.4, and A5.5 are identical. When version-specific features are discussed in this chapter for these versions, they are noted as *pre-OpenVMS 5.5*. Some features are specific to OpenVMS VAX 5.5, OpenVMS AXP 1.5, and OpenVMS VAX 6.0; these are noted as *post-OpenVMS 5.5*. Some features are specific to OpenVMS 6.0 only, and are indicated as such. Finally, any features discussed in this chapter that are not specifically noted as version-specific are applicable to *all* OpenVMS versions.

Table 8.1 Summary of Queue Management Changes

Feature	*OpenVMS V5.4, A5.5 OpenVMS AXP 1.0 pre-OpenVMS 5.5*	*OpenVMS V5.5 OpenVMS AXP 1.5 post-OpenVMS 5.5*	*OpenVMS 6.0 post-OpenVMS 5.5*
Queue Manager	Distributed	Centralized	Centralized
Queue Manager Fail-over	No	Yes	Yes
Queue Autostart and Fail-over	No	Yes	Yes
Multiple Queue Managers	No	No	Yes

8.3 Queue Types

The OpenVMS queue management subsystem allows printers and computer resources to be shared using two types of queues, as follows.

Print Queues

To share a printer among several users or systems, establish a *print queue* and associate it with a physical printer. After users submit jobs (one or more files) to the queue, the jobs are sent to the associated printer. In this way, print queues allow multiple users to simultaneously submit jobs to a single printer and continue working on other tasks while the job prints.

Print queues can be designated as either *generic* or *output*. An output print queue has an associated physical printer to which jobs are directed. A generic print queue serves only as a logical queue and directs jobs to one or more associated output queues. You can use generic queues to load-balance print jobs among several printers and as a method to provide better printer availability.

For example, if you had three LP37 line printers in a shipping department, you could configure a single generic queue to feed three output queues as follows:

```
LP37_SHIPPING                                  (Generic Print Queue)
|                                Device
=>   LP37_SHIPPING_1 =>          TXA0:         (Output Print Queue)
=>   LP37_SHIPPING_2 =>          TXA1:                  "
=>   LP37_SHIPPING_3 =>          TXA2:                  "
```

When many print jobs are submitted to the generic queue (LP37_SHIPPING), they are evenly distributed for printing to the three output queues. If one of the printers fails, print jobs are directed to the remaining printers.

Printed output is frequently critical to the operation of an organization (e.g., checks, documents, invoices and reports). Within this context, some printer failures can have a business impact that is as great as a complete system failure. As a result, you should give careful consideration to the printer availability requirements at your site and implement an appropriate print queue strategy.

Batch Queues

Batch queues provide a mechanism to schedule, coordinate, and share access to the compute resources of a system. Users or applications submit jobs (DCL command procedures) to a batch queue. When a job is ready for execution, a process is created, and the job is executed on the system associated with the queue.

Batch queues can be designated as either *generic* or *execution* queues. Jobs submitted to an execution queue are directed to execute on the system designated by the queue. Generic batch queues direct submitted jobs to one or more associated execution queues. In this way, generic batch queues provide load-balancing and high-availability capabilities for batch execution queues.

For example, consider a VMScluster (FOREST), consisting of three VAX 6410 systems (BIRCH, LOTUS, and HEMLCK). You could configure a cluster-wide generic batch queue (FOREST_GENERAL) to direct jobs to separate execution queues on each cluster member, as follows:

```
FOREST_GENERAL              System          (Generic Batch Queue)
|
=>    BIRCH_GENERAL =>      BIRCH::         (Execution Batch Queue)
=>    LOTUS_GENERAL =>      LOTUS::                  "
=>    HEMLCK_GENERAL =>     HEMLCK::                 "
```

Batch jobs submitted to FOREST_GENERAL are directed to the execution queue on the system that is least busy. If one system fails, batch jobs can continue to be processed by the remaining execution queues.

Depending on your system configuration, you may use applications that rely heavily on batch queues. The failure of a single batch queue can render an application unavailable, even though the system as a whole is operational. As with print queues, you should carefully consider the batch queue availability requirements of your site.

Figure 8.1 Pre-OpenVMS 5.5 Queue Manager

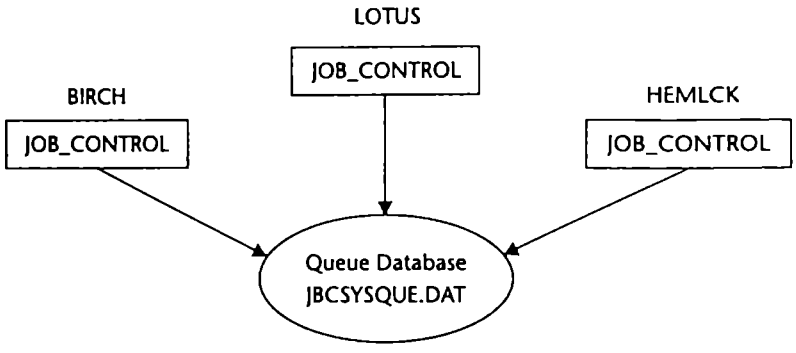
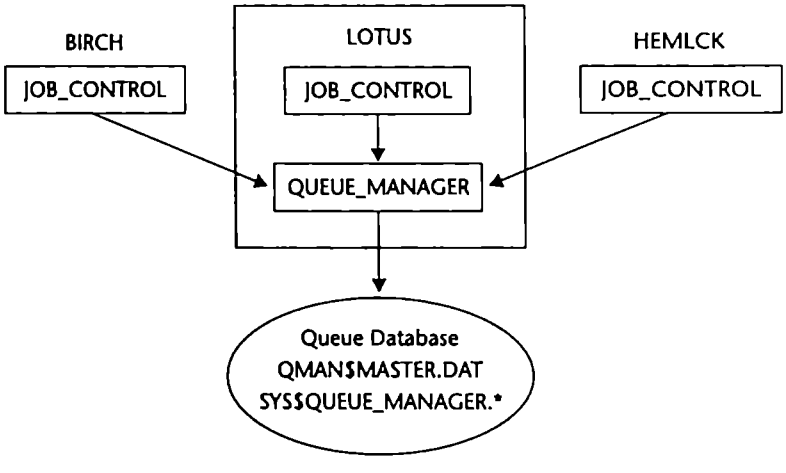


Figure 8.2 Post-OpenVMS 5.5 Queue Manager



8.4 Queue Manager

The Queue Manager handles batch and print queue management functions, such as creating, starting, and stopping queues and manipulating jobs.

[pre-OpenVMS 5.5]

In versions of OpenVMS prior to version 5.5, the Queue Manager is designed using a *distributed* model, in which all queue management functions are performed by one or more JOB_CONTROL processes. On a stand-alone system

Table 8.2 Queue Database Files

Database Item	*pre-OpenVMS 5.5*	*post-OpenVMS 5.5*
Defined queue names	JBCSYSQUE.DAT	QMAN$MASTER.DAT
Form definitions	JBCSYSQUE.DAT	QMAN$MASTER.DAT
Queue characteristics	JBCSYSQUE.DAT	SYS$QUEUE_MANAGER.QMAN$QUEUES
Job Information	JBCSYSQUE.DAT	SYS$QUEUE_MANAGER.QMAN$JOURNAL
Queue and journal names	N/A	QMAN$MASTER.DAT
Lists of fail-over nodes	N/A	QMAN$MASTER.DAT
Fail-over/recovery data	N/A	SYS$QUEUE_MANAGER.QMAN$JOURNAL

the JOB_CONTROL process performs queue management functions. In a common-environment VMScluster, queue management functions are distributed among the JOB_CONTROL processes on each cluster member, although each accesses a common queue database file, SYS$COMMON:[SYSEXE] JBCSYSQUE.DAT. Because multiple processes have to coordinate access to the queue database, performance degrades under heavy queue activity.

[post-OpenVMS 5.5]

With the introduction of OpenVMS version 5.5, the Queue Manager is designed using a centralized model consisting of a single, cluster-wide QUEUE_MANAGER process that performs all queue management functions. Users and the JOB_CONTROL processes on each system communicate with the QUEUE_MANAGER process by means of a shared Inter-Process Communication (IPC) link. Because only the single QUEUE_MANAGER process accesses the queue database, contention and locking overhead are minimized, allowing better performance within environments with heavy queue activity.

8.4.1 Queue Database

The queue database stores queue and form definitions, job information, and fail-over information (post-OpenVMS 5.5). This data is stored in different files depending on the OpenVMS version, as indicated in Table 8.2.

Within a common-environment VMScluster, a single copy of the files in Table 8.2 is normally shared among cluster members and located in the directory SYS$COMMON:[SYSEXE]. Within a mixed-environment VMScluster,

each cluster member typically maintains its own copy of the files, each system locating its copy in SYS$SPECIFIC:[SYSEXE].

[OpenVMS 6.0]

Beginning with OpenVMS 6.0, you can have multiple queue managers, which results in additional queue and journal files. For example, if you added a queue manager named PRINT_MGR, two additional files are created to contain queue and job information for that queue manager, as follows:

```
PRINT_MGR.QMAN$QUEUES     ! Queue characteristics
PRINT_MGR.QMAN$JOURNAL    ! Job characteristics
```

8.5 Controlling the Queue Manager

Before any print and batch queue functions can be initiated, the Queue Manager must be started.

[pre-OpenVMS 5.5]

If your version is earlier than OpenVMS 5.5, you start the queue manager each time you boot the system by including the following command in system startup procedures:

```
$ START/QUEUE/MANAGER
```

This command assumes that queue database files are located in the default directory (SYS$SYSTEM:).

If your system disk is heavily utilized, you can locate the queue database files in an alternative directory by specifying the directory name when you start the Queue Manager, as follows:

```
$ START/QUEUE/MANAGER DISK$USER1:[SYSFILES]JBCSYSQUE.DAT
```

[post-OpenVMS 5.5]

Beginning with OpenVMS 5.5, you need to start the Queue Manager only once, using the following command:

```
$ START/QUEUE/MANAGER
```

Once the Queue Manager is started, its state is recorded in the queue database. When you boot the system, OpenVMS checks the previous state of the Queue Manager and automatically returns it to its previous state. Thus, as long as the Queue Manager is started when you perform a system shutdown, you need not include a START/QUEUE/MANAGER command in system startup procedures.

If your system disk is heavily utilized, you can move the Queue Manager master file to another location by defining a system logical name as follows:

```
$ DEFINE/SYSTEM/EXEC QMAN$MANAGER DISK$USER1:[SYSFILES]
```

To make this change permanent, add the above command to the startup procedure SYLOGICALS.COM, as discussed in chapter 5.

The queue and journal files can be located in an alternative directory (not necessarily the same location as the master file). This is accomplished by adding a directory specification when you start the Queue Manager, as follows:

```
$ START/QUEUE/MANAGER DISK$USER1:[SYSFILES]
```

Once you issue the above command, the directory you specify is permanently stored in the queue master file; you don't have to specify it again. Table 8.3 summarizes the DCL commands you use to start and stop the Queue Manager.

8.5.1 Queue Manager Fail-over

[post-OpenVMS 5.5]

If the system operating as the Queue Manager fails, is removed from a cluster, or is stopped, all queues serviced by that Queue Manager are stopped—unless it is configured for automatic fail-over. You ensure proper fail-over by specifying a fail-over candidate list when you start the Queue Manager; for example:

```
$ START/QUEUE/MANAGER/ON=(BIRCH,LOTUS)
```

In the example, the preferred Queue Manager node is designated as BIRCH with LOTUS as a fail-over candidate. You can specify that all members of a VMScluster are valid fail-over candidates, as follows:

```
$ START/QUEUE/MANAGER/ON=(*)
```

Table 8.3 Queue Manager Command Summary

DCL Command	*Pre-OpenVMS 5.5 Behavior*	*Post-OpenVMS 5.5 Behavior*
`START/QUEUE/MANAGER/ NEW_VERSION`	Create a new queue database Use this command with extreme care because creating a new queue database deletes any previously defined queues, forms, or printer characteristics	Same
`START/QUEUE/MANAGER`	Starts the Queue Manager for the first time	Same
`STOP/QUEUE/MANAGER`	Stops the Queue Manager	Stops all queues on the local system
`STOP/QUEUE/MANAGER/ CLUSTER`	N/A	Stops the Queue Manager (even on nonclustered systems)

You may also want to specify a preferred Queue Manager *and* specify that remaining cluster members act as fail-over candidates, as follows:

```
$ START/QUEUE/MANAGER/ON=(BIRCH,*)
```

8.5.2 Multiple Queue Managers

[OpenVMS 6.0]

In environments where there is an extremely high job submission rate (e.g., many jobs per second), you may want to consider creating multiple queue managers. Beginning with OpenVMS 6.0, you can establish up to five queue managers per cluster. This allows you to spread the overhead generated by the Queue Manager among several VMScluster members. For example, in a two-node cluster consisting of members BIRCH and LOTUS, you could establish the default queue manager on BIRCH to handle all cluster batch queues, then create a second queue manager on LOTUS to handle all cluster print queues.

The following command establishes a second queue manager named PRINT_MGR on node LOTUS:

```
$ START/QUEUE/MANAGER/ADD/NAME=PRINT_MGR/ON=(LOTUS::,*) DISK$USER1:[SYSFILES]

$ SHOW QUEUE/MANAGER/FULL
Master file: DISK$USER1:[SYSFILES]QMAN$MASTER.DAT;

Queue manager PRINT_MGR, running, on LOTUS::
   /ON=(LOTUS,*)
   Database location: DISK$USER1:[SYSFILES]

Queue manager SYS$QUEUE_MANAGER, running, on BIRCH::
   /ON=(BIRCH,*)
   Database location: DISK$USER1:[SYSFILES]
```

To create queues on a queue manager other than the default, add the /NAME_OF_MANAGER qualifier to queue initialization commands, for example:

```
$ INITIALIZE/QUEUE/ON=LOTUS::TXA1:/NAME=PRINT_MGR SALES_LN03_2
```

The following restrictions apply if you use multiple queue managers:

- All cluster members must be running OpenVMS Version 6.0
- Queues running on one manager cannot reference queues running on a different queue manager. For example, generic queues and their associated execution queues must be started on the same queue manager. Also, you cannot re-queue a job from one queue manager to another

8.6 Queue Naming Conventions

You should carefully develop a queue naming convention *before* you begin configuring queues. Frequently, queue names are assigned with the intent of renaming them later. However, you quickly find that so many users depend on the temporary names that changing them is impossible.

Queue names should be consistent, and it should be easy for users to associate a queue name with a queue. With printers, users are primarily concerned with the printer type and its location. Therefore, one possible strategy is to form printer queue names with the printer *type* and its *location*, separated by an underscore (_) as follows:

```
LOCATION_TYPE
```

You may want to specify *location* in terms of a department name, rather than a physical building or room identifier. For example, use ACCT, HR, and SHIP to specify the departments accounting, human resources, and shipping, respectively. Choose a naming convention that will not have to change as people, departments, or equipment move. Otherwise you'll spend all your time changing queue names. You can specify printer *type* in terms of the printer make or model (e.g., LN03, HPIIP, LG31), or in a more general sense (e.g., LASER, LINE, or LABEL).

When you have multiple printers at the same location, add a sequential number or letter at the end of the queue name. If a generic queue feeds several output queues, give it the same name as the output queues, but omit the letter or number at the end of the name. For example

```
BILLING_LP37_A      ! LP37 Line Printer in Billing Dept.
SALES_LN03_2        ! LN03 Laser Printer in Sales Dept.
HR_HPLASER_1        ! first HP Laser Printer in Human Resources
HR_HPLASER_2        ! second HP Laser Printer in Human Resources
HR_HPLASER          ! Human Resources generic queue
```

With batch queues, one possible strategy is to define queue names based on the *function* of the queue and its associated *node name*, as follows:

```
NODENAME_FUNCTION
```

Function might specify GENERAL, BACKUPS, BIG, MEDIUM, or SMALL for general jobs, system backups, or big, medium, and small batch jobs, respectively. For example

```
LOTUS_BACKUPS       ! Queues on LOTUS:: for running system backups
BIRCH_BIG           ! Batch queue for BIG jobs on node BIRCH::
LOTUS_GENERAL       ! General Batch queue on node LOTUS::
FOREST_GENERAL      ! Cluster-wide generic queue
```

8.7 Configuring Printer Devices

Before a print output queue can be configured, the device to which it will send output must be configured. Print output queues can be associated with one of the following devices:

- Terminal device (e.g., TXA0:, TTA0:, etc)
- Local area transport device (e.g., LTA1:, LTA2:, etc)
- Printer device (e.g., LPA0:, etc)

8.7.1 Terminal Device

Before you can use a terminal device with a print queue, you have to configure it using the SET TERMINAL command, as follows:

```
$ SET TERMINAL/qualifiers device-name
```

You may need to specify the following qualifiers:

```
/DEVICE=device-type
       (e.g., LA34, LA100, LA210, LN03, UNKNOWN)

/SPEED=baud-rate
       (e.g., 1200, 2400, 4800, 9600)

/NOBROADCAST              ! Disable system broadcast messages
/PAGE=page-length         ! Set lines per page
/WIDTH=page-width         ! Set characters per line
/NOTYPE_AHEAD             ! Disables interactive logins
/WRAP                     ! Wrap lines > page-width to next line
/NOWRAP                   ! Truncate lines > page-width
/PERMANENT                ! Make settings permanent
```

Each *device-type* has associated default characteristics for /PAGE, /WIDTH, etc. When you issue the SET TERMINAL/DEVICE command, any characteristics you do not specify are taken from the device defaults (see SET TERMINAL documentation in the *OpenVMS DCL Dictionary* for more information). For example:

```
$ SET TERMINAL/DEVICE=LA120 TXA0:   ! Defaults to /WIDTH=132
$ SET TERMINAL/DEVICE=LN03 TXA0:    ! Defaults to /WIDTH=80
```

You should set /NOBROADCAST on all terminal devices used for print queues; otherwise, system broadcast messages are printed on the printer. For security purposes, you may also want to set /NOTYPE_AHEAD. This prevents someone from replacing a printer with a terminal and attempting an interactive login.

Once you have determined the appropriate SET TERMINAL commands, add them to your site-specific startup procedures. You should also add the /PERMANENT qualifier so that terminal settings stay in effect until the system is re-booted. As discussed in chapter 5, place these commands in a command procedure like STARTUP_SETTERM.COM. For example

```
$! STARTUP_SETTERM.COM
$!...
```

```
$ SETTERM_PRT:=SET TERMINAL/PERMANENT/NOBROADCAST/WRAP/NOTYPE_AHEAD
$ SETTERM_PRT/SPEED=9600/DEVICE=LN03     TXA0:
$ SETTERM_PRT/SPEED=9600/DEVICE=LA100    TXA1:
```

8.7.2 LAT Device

Print queues can be configured to utilize printers attached to ports on terminal servers using the LAT protocol. To configure LAT devices, perform the following procedure.

Configure the Terminal Server Port

The port on the terminal server must be configured to be used by a printer rather than an interactive terminal. To configure a terminal server port, you must have access to an interactive terminal attached to the server, or initiate a connection to the server maintenance port using the OpenVMS Network Control Program (NCP).

To access the server from NCP, issue the following commands:

```
$ RUN SYS$SYSTEM:NCP      ! Run Network Control Program (NCP)
NCP> CONNECT NODE TS001   ! TS001 = terminal server name
#                         ! enter terminal server password
                          ! by default, "ACCESS"
```

If you have a locally attached terminal, start from this point:

```
Local>                    ! Enter Local> mode

Local> SET PRIVILEGE      ! Enable privileged mode
Password:                 ! enter privilege password
                          ! by default, "SYSTEM"
Local>
```

To connect to a server, NCP must know the server's Ethernet hardware address (the twelve-digit number printed on the server chassis). If the server is not defined, you can define it manually, as follows:

```
$ RUN SYS$SYSTEM:NCP
NCP> DEFINE NODE 1.100 NAME TS001 HARDWARE ADDR 08-00-2B-A1-83-BE
NCP> SET NODE 1.100 ALL
```

If you prefer not to define the node, you can explicitly specify the hardware address with the connect command, as follows:

```
NCP> CONNECT VIA circuit PHYSICAL ADDRESS 08-00-2B-A1-83-BE
```

(Where *circuit* is the name of the Ethernet circuit for the system, e.g., BNA-0, QNA-0, etc.). Next, configure the ports to which you want to connect a printer as follows:

```
Local> DEFINE PORT 1 ACCESS REMOTE ENABLED SPEED 4800 -
       BROADCAST DISABLE NAME "PORT_1"

Local> DEFINE PORT 2 ACCESS REMOTE ENABLED SPEED 4800 -
       BROADCAST DISABLE NAME "PORT_2"

Local> LOGOUT PORT 1      ! copy permanent to volatile database
Local> LOGOUT PORT 2      ! copy permanent to volatile database
```

The following describes each of the foregoing settings:

```
REMOTE ENABLED            ! Configures port for printer use
SPEED 4800                ! Set device speed to 4800 baud
BROADCAST DISABLE         ! Disable system broadcast messages
NAME  "                   ! Name used to reference server port
```

If you plan to establish print queues on multiple systems that will all reference the same terminal server port, you should add the QUEUE ENABLE option to the foregoing defining statements. Terminal server port settings are saved even if the terminal server loses power. However, ports must be re-configured if the server is replaced or reset to factory settings.

Create LAT Device

Next, use the LAT Control Program (LATCP) to create a LAT device that points to the desired terminal server and port, as follows:

```
$ RUN SYS$SYSTEM:LATCP
LATCP> CREATE PORT LTA100:
LATCP> MODIFY PORT LTA100:/NODE=TS001/PORT="PORT_1"
LATCP> CREATE PORT LTA101:
LATCP> MODIFY PORT LTA101:/NODE=TS001/PORT="PORT_2"
```

Configure Device

Finally, use SET TERMINAL to configure other device characteristics not controlled at the terminal server level

```
$ SETTERM_PRT:=SET TERMINAL/PERMANENT/NOBROADCAST/WRAP/NOTYPE_AHEAD
$ SETTERM_PRT/DEVICE=LN03      LTA100:
$ SETTERM_PRT/DEVICE=LA100     LTA101:
```

As discussed in chapter 5, these commands should be added to site-specific startup procedures (e.g., STARTUP_SETTERM.COM).

8.7.3 Printer Device

When print queues are established to printer devices, such as LPA0, configure the device using the SET PRINTER command, as follows:

```
$ SET PRINTER/qualifiers printer-device
```

You may need to specify one or more of the following qualifiers:

Printer type

```
/LP11                   ! LP printer type (default)
/LA11                   ! LA printer type
/LA180                  ! LA180 printer
/UNKNOWN                ! Other printer type
```

Page formatting

```
/PAGE=page-length       ! Set lines per page (default = 64)
/WIDTH=page-width       ! Set characters per line (default = 132)
/UPPERCASE              ! Print all uppercase letters (default)
/NOUPPERCASE            ! Print upper and lowercase
/WRAP                   ! Wrap lines > page-width to next line
/NOWRAP                 ! Truncate lines > page-width (default)
```

For example, to configure printer device LPA0, issue the following command:

```
$ SET PRINTER/PAGE=60/WIDTH=132/WRAP LPA0:
```

As discussed in chapter 5, add the foregoing commands to site-specific startup procedures (e.g., STARTUP_SETTERM.COM).

8.7.4 Enabling Spooling

When you establish a print queue and associate it with a device, all output sent to the device should originate from the queue. If you attempt to bypass the queue and send output directly to the device, the queue may fail. If you have applications that send printer output directly to a device that has an associated print queue, you should enable spooling on the device, as follows:

```
$ SET DEVICE/SPOOL=(device, print-queue) device-name
```

For example,

```
$ SET DEVICE/SPOOL=(SYS$SYSDEVICE, SALES_LN03_2) TXA0:
```

When output is sent directly to the device TXA0:, the output is redirected to a temporary spool file on the *device* specified. When output is completed, the spool file is submitted as a job to the *print-queue* specified. In other words, spooling forces all output to a print queue. Add all spool commands to site-specific startup procedures (e.g., STARTUP_SETTERM.COM).

8.8 Creating and Starting Queues

To create and start queues, use the INITIALIZE/QUEUE and START/QUEUE commands. Beginning with OpenVMS 6.0, specify the /NAME_OF_MANAGER command to establish or start a queue associated with an alternative queue manager.

The following sections discuss how to create and start various batch and output queues.

8.8.1 Autostart Execution Queues

[Post-OpenVMS 5.5]

Beginning with OpenVMS 5.5, execution queues can be configured as either autostart or nonautostart. The new Autostart feature allows you to start and stop all autostart enabled queues with a single command, as follows:

```
$ ENABLE AUTOSTART/QUEUES
$ DISABLE AUTOSTART/QUEUES
```

This feature greatly simplifies system startup procedures. Previously, you had to include commands in system startup procedures to start each queue. Using the Autostart feature, you can replace hundreds of INITIALIZE/QUEUE/START statements with a single ENABLE AUTOSTART/QUEUES command.

Autostart queues also provide enhanced fail-over capabilities. Autostart batch queues can be configured to fail over to an alternative VMScluster member if the current member fails. Similarly, autostart print queues that use LAT devices can be configured to fail over to an alternative system and device if the current system fails. To take advantage of these features you should configure queues for autostart whenever possible.

The following command creates an autostart print queue on node BIRCH:

```
$ INITIALIZE/QUEUE/AUTOSTART_ON=BIRCH::TTA1: SALES_LN03_2
```

The following command creates an autostart batch queue on node BIRCH:

```
$ INITIALIZE/QUEUE/BATCH/AUTOSTART_ON=BIRCH:: BIRCH_GENERAL
```

Once you create a queue, its configuration is stored in the queue database—you don't have to execute the INITIALIZE command in system startup procedures.

8.8.1.1 Starting Autostart Queues

Autostart queues are automatically started during system startup. However, if you manually stop a queue, you can start it with the following command:

```
$ START/QUEUE queue-name
```

To manually restart all autostart queues enter the following command:

```
$ ENABLE AUTOSTART/QUEUES
```

8.8.2 Standard (Nonautostart) Execution Queues

Nonautostart queues are configured nearly the same as autostart queues, except use the /ON qualifier in place of the /AUTOSTART_ON qualifier. In addition, nonautostart queues must be individually started with a START/QUEUE or INITIALIZE/QUEUE/START command and must be explicitly started during system startup. Because this process increases the complexity of system startup procedures, you should use standard queues only when you don't want queues to start automatically during system startup.

For example, to create a nonautostart *batch* queue (BIRCH_GENERAL), issue the following command:

```
$ INITIALIZE/QUEUE/BATCH/ON=BIRCH:: BIRCH_GENERAL
```

Once you create the queue, start it with the following command:

```
$ START/QUEUE BIRCH_GENERAL
```

To automatically start the queue after a system re-boot, add the above command to site-specific startup procedures (e.g., STARTUP_QUEUES.COM). You can optionally add the /START qualifier to the INITIALIZE/QUEUE command, which eliminates the need for the START/QUEUE command:

```
$ INITIALIZE/QUEUE/BATCH/START/ON=BIRCH:: BIRCH_GENERAL
```

To create a nonautostart *print* queue use the INITIALIZE/QUEUE/ON= command. For example, to create a queue SALES_LN03_2 on device TXA1 use the following:

```
$ INITIALIZE/QUEUE/ON=BIRCH::TXA1: SALES_LN03_2
```

Once the queue is created, start it with the following command:

```
$ START/QUEUE SALES_LN03_2
```

To start the queue automatically after a system re-boot, add the above command to site-specific startup procedures (e.g., STARTUP_QUEUES.COM). You can optionally add the /START qualifier to the INITIALIZE/QUEUE command, which eliminates the need for the START_QUEUE command:

```
$ INITIALIZE/QUEUE/START/ON=BIRCH::TXA1: SALES_LN03_2
```

8.8.3 Generic Queues

Generic queues allow you to load-balance print and batch jobs among several printers or systems. Normally, you must use the Autostart feature to configure queues for automatic fail-over. However, if you cannot configure a queue as autostart, or are *not* running OpenVMS Version 5.5 (or higher), you can achieve some level of queue fail-over capability by using generic queues.

8.8.3.1 Load-balancing Print Queues

For example, to load-balance incoming jobs to three LP37 printers in a shipping department, create the following queues:

```
$ INITIALIZE/QUEUE/START/ON=BIRCH::TXA1: LP37_SHIPPING1
$ INITIALIZE/QUEUE/START/ON=BIRCH::TXA2: LP37_SHIPPING2
$ INITIALIZE/QUEUE/START/ON=BIRCH::TXA3: LP37_SHIPPING3

$ INITIALIZE/QUEUE/START -
/GENERIC=(LP37_SHIPPING1, LP37_SHIPPING2, LP37_SHIPPING3) LP37_SHIPPING
```

8.8.3.2 Load-balancing Batch Queues

For example, to load-balance batch jobs among three VMScluster members, perform the following:

```
$ INITIALIZE/QUEUE/BATCH/START/ON=BIRCH:: BIRCH_GENERAL
$ INITIALIZE/QUEUE/BATCH/START/ON=LOTUS:: LOTUS_GENERAL
$ INITIALIZE/QUEUE/BATCH/START/ON=HEMLCK:: HEMLCK_GENERAL

$  INITIALIZE/QUEUE/START -
/GENERIC=(BIRCH_GENERAL,LOTUS_GENERAL,HEMLCK_GENERAL) FOREST_GENERAL
```

8.8.3.3 Automatic Fail-over with Nonautostart Queues

To create a cluster-wide print queue with automatic fail-over, create the following queues:

```
$ INITIALIZE/QUEUE/PROCESSOR=LATSYM/ON=(BIRCH::LTA100:) SALES_LN03_2_BIRCH

$ INITIALIZE/QUEUE/PROCESSOR=LATSYM/ON=(LOTUS::LTA100:) SALES_LN03_2_LOTUS

$ INITIALIZE/QUEUE/PROCESSOR=LATSYM/ON=(HEMLCK::LTA100:) SALES_LN03_2_HEMLCK

$ INITIALIZE/QUEUE/START/GENERIC= -
(SALES_LN03_2_BIRCH, SALES_LN03_2_LOTUS, SALES_LN03_2_HEMLCK) SALES_LN03_2
```

8.8.3.4 Automatic Fail-over with Autostart Queues

Configuring automatic fail-over with autostart queues is much simpler than with nonautostart queues. The following single queue accomplishes the same level of fail-over that requires four queues when nonautostart queues are used:

```
$ INITIALIZE/QUEUE/PROCESSOR=LATSYM/AUTOSTART_ON= -
      (BIRCH::LTA100:, LOTUS::LTA100:, HEMLCK::LTA100:) SALES_LN03_2
```

8.8.4 Default Print and Batch Queue Logical Names

By default, print and batch jobs are submitted to the queue specified by the logical names SYS$PRINT and SYS$BATCH. Do not create queues named SYS$PRINT and SYS$BATCH. Use queue names as described in this chapter and define logical names to specify default system-wide queues as follows:

```
$ DEFINE/SYSTEM/EXEC SYS$BATCH BIRCH_GENERAL
$ DEFINE/SYSTEM/EXEC SYS$PRINT SALES_LN03_2
```

The procedures for defining these logical names during system startup and system login are discussed in chapters 5 and 11.

8.9 Other Queue Characteristics

In addition to the qualifiers already discussed, several other qualifiers may be specified when creating queues. Some qualifiers are specific to batch queues, others to print queues, and the rest are applicable to both, as illustrated in Table 8.4.

These qualifiers are described in more detail in *Guide to Maintaining an OpenVMS System (V5.5)*, *OpenVMS Systems Manager's Manual: Essentials*, and *OpenVMS DCL Dictionary.*

8.10 Device Control Libraries

Use device control libraries to include special character sequences (escape sequences) at the beginning or end of a print job. These characters configure printer characteristics, such as

- Page orientation (landscape or portrait)
- Printer font (Times Roman, Courier, etc.)
- Character point size
- Character lines per inch

A device control library contains a series of *setup modules*. Each setup module is a text file that contains escape sequences to perform certain printer functions. When you want to activate a printer function you request that the setup module be sent to the printer.

8.10.1 Create Setup Modules

You create a setup module by creating text files that contain printer programming sequences

```
$ SET DEFAULT DISK$USER1:[SYSFILES.SETUP.HP]

$ EDIT LANDSCAPE.TXT      ! HP Laser Jet Landscape (see HP Manual)

$ EDIT PORTRAIT.TXT       ! HP Laser Jet Portrait (see HP Manual)

$ EDIT SOFT_RESET.TXT     ! HP Laser Jet Reset (see HP Manual)
```

Table 8.4 Print and Batch Queue Characteristics

Qualifier	*Function*
BATCH and PRINT	
/BASE_PRIORITY	Establishes base priority for batch jobs or base priority of symbiont process (print queues)
/WSDEFAULT /WSQUOTA /WSEXTENT	Specifies working set parameters for batch jobs or for symbiont process (print queues)
/CHARACTERISTICS	Associates printer characteristics with queue
/DESCRIPTION	Uses text to describes queue
/RETAIN	Retains jobs in queue after processing
/OWNER_UIC	Displays owner user identification code of queue (see chapter 17)
/PROTECTION	Provides protection mask on queue (see chapter 17)
BATCH Only	
/CPUDEFAULT	Defaults maximum central processing unit (CPU) time a job can use
/CPUMAXIMUM	Shows maximum CPU time a job can request
/DISABLE_SWAPPING	Prevents batch jobs from swapping
/JOB_LIMIT	Limits number of concurrent jobs being executed
PRINT Only	
/BLOCK_LIMIT	Shows maximum allowed print job size
/DEFAULT	Defaults flag, form, and burst pages
/DEVICE	Specifies either Printer, Server, or Terminal
/FORM_MOUNTED	Shows mounted form being used
/LIBRARY	Provides device control library containing setup modules
/PROCESSOR	Shows printer symbiont type
/RECORD_BLOCKING	Sends individual job records in larger blocks
/SEPARATE	Controls separation pages between files and jobs
/SCHEDULE	Allows scheduling of jobs by size

```
$ SET DEFAULT DISK$SYS1:[SYSFILES.SETUP.LN03]

$ EDIT LANDSCAPE.TXT      ! LN03 - Select Landscape Orientation
<ESC>]?21 J

$ EDIT PORTRAIT.TXT       ! LN03 - Select Portrait Orientation
<ESC>]?20 J

$ EDIT LN03_RESET.TXT     ! LN03 - Reset
<ESC>C
```

The *escape* character (<ESC>) is entered by using the special insert function within the EDT Editor by pressing the following key sequence:

```
[GOLD]27[GOLD][KEYPAD-3]
```

8.10.2 Creating a Device Control Library

Create a separate device control library for each printer type, as follows:

```
$ LIBRARY/CREATE/TEXT SYS$LIBRARY:HP_DEVCTL.TLB
$ LIBRARY/CREATE/TEXT SYS$LIBRARY:LN03_DEVCTL.TLB
```

8.10.3 Adding Modules

Add the modules, as follows:

```
$ LIBRARY/INSERT/TEXT SYS$LIBRARY:HP_DEVCTL.TLB -
  DISK$USER1:[SYSFILES.SETUP.HP]*.TXT

$ LIBRARY/INSERT/TEXT SYS$LIBRARY:LN03_DEVCTL.TLB -
  DISK$USER1:[SYSFILES.SETUP.LN03]*.TXT
```

8.10.4 Using Device Control Libraries

Before a setup module can be sent to a print queue, the queue must be assigned to a device control library using the /LIBRARY qualifier, as follows:

```
$ INITIALIZE/QUEUE/LIBRARY=LN03_DEVCTL/SEPARATE=RESET=SOFT_RESET SALES_LN03_2
```

The /SEPARATE qualifier forces the SOFT_RESET module to be sent to the printer after each print job. This forces the printer to reset to its default configuration, preventing one job from setting printer characteristics that conflict with future jobs. Once the queue is configured properly, include a setup module when printing a job, as follows:

```
$ PRINT/SETUP=setup-module filename
```

For example, to print a file in landscape mode

```
$ PRINT/SETUP=LANDSCAPE filename
```

You can also associate a set module with a form definition, as follows:

```
$ DEFINE/FORM LANDSCAPE 1/SETUP=LANDSCAPE
```

When you submit a print job with a form that has an associated setup module, the module is sent to the printer before the job, as follows:

```
$ PRINT/FORM=LANDSCAPE filename
```

8.11 Print Forms

The purpose of print forms is to allow the following functions:

- Format page output (margins, page length, and page width)
- Facilitate printing on multiple paper stock

8.11.1 Creating Forms

Forms are defined using the following command:

```
$ DEFINE/FORM/qualifiers form-name form-number
```

A form has the following characteristics:

```
form-name           ! form name ( 1-31 characters long)

form-number         ! form number (0 - 2 million)

/MARGIN=
   BOTTOM=lines     ! Bottom margin
   TOP=lines        ! Top margin
   LEFT=characters  ! Left margin
   RIGHT=characters ! Right margin

/STOCK=stock-name   ! Paper stock that form requires
                    ! (e.g., 8_5X11, 11X14)
/WRAP               ! if (line-width > page-width) wrap to next line
/NOWRAP             ! if (line-width > page-width) truncate

/WIDTH=page-width   ! Page width

/LENGTH=page-length ! Page length

/SETUP=setup-module      ! Setup-module to send before each job

/PAGE_SETUP=setup-module ! Setup-module to send before each page
```

```
/SHEET_FEED          ! For sheet-fed printers only

/DESCRIPTION="text"  ! Description of form
```

You can view the forms defined on your system by issuing the following command:

```
$ SHOW QUEUE/FULL/ALL/FORM
```

Digital supplies the system default form (DEFAULT) which has the following characteristics:

```
Form name            Number          Description
---------            ------          -----------
DEFAULT              0               System-defined default
   /MARGIN=(BOTTOM=6) / STOCK=DEFAULT /TRUNCATE /WIDTH=132
   /LENGTH=66
```

The following commands are used to create several example forms:

```
$ DEFINE/FORM LETTERHEAD 2 -
/STOCK=8_5X11/MARGIN=(TOP=10,BOTTOM=5,LEFT=10)/WIDTH=80/LENGTH=66)-
/NOWRAP/DESCRIPTION="Corporate Letterhead"

$ DEFINE/FORM PLAIN 3 -
/STOCK=8_5X11/MARGIN=(TOP=2,LEFT=5)/WIDTH=80/LENGTH=66)/NOWRAP -
/DESCRIPTION="Plain paper"

$ DEFINE/FORM LABEL1 4 -
/STOCK=1X4LABEL/MARGIN=(TOP=1,LEFT=1)/WIDTH=40/LENGTH=6) -
/NOWRAP/DESCRIPTION="1 x 4 Label"
```

When you define forms, specify a stock name that represents the physical type of paper used (e.g., 8_5X11).

8.11.2 Printing Jobs Using Forms

When you create a queue, you establish the *mounted form* for the queue using the following command:

```
$ INITIALIZE/QUEUE/FORM_MOUNTED=form-name queue-name
```

By default, the mounted form for a queue is the *system default form* (named DEFAULT). A queue also has an associated *mounted stock* that matches the

stock of the mounted form. By default, the DEFAULT form has an associated stock also named DEFAULT.

To specify a different default mounted form (and stock) for a queue, create the queue as follows:

```
$ INITIALIZE/QUEUE/DEFAULT=FORM=form-name queue-name
```

When you submit a print job, specify its *printed form* using the following command:

```
$ PRINT/FORM=form-name filename
```

When a job is submitted to a queue without a specified printed form, the printed form for the job is set to the default form for the queue. Print jobs are accepted into a queue regardless of their printed form. However, a queued job can print only when the *stock* of the printed form matches the *stock* of the mounted form. When there is a stock mismatch, the queued job remains in a pending state until one of the following conditions occurs:

1. The mounted form for the queue is changed (after the correct paper stock is loaded) to a form with a stock that matches the stock for the print job, as follows:

   ```
   $ SET QUEUE/FORM_MOUNTED=form-name queue-name
   ```

2. The printed form for the job is changed to a form with stock that matches the stock of the mounted form, as follows:

   ```
   $ SET ENTRY/FORM=form-name entry-number
   ```

8.12 Additional References

Building Dependable Systems, Digital, November 1991.
Guide to Maintaining an OpenVMS System, Digital, November 1991.
VAXcluster Manual, Digital, November 1991.

[new V6.0]

OpenVMS DCL Dictionary, Digital, March 1994.
OpenVMS Systems Manager's Manual: Essentials, Digital, March 1994.

9 MANAGING USER ACCOUNTS

9.1 Chapter Objectives

- Standardize username formats
- Establish account groups to facilitate access control
- Ensure minimum password content standards
- Create templates for frequently added accounts
- Automate account management functions

9.2 Introduction

The *user account* is the primary mechanism that the system uses to control system access and resource usage. Before a user can access the system, you must create an account for the user and assign the account sufficient resources for the user's applications.

9.3 User Authorization Database

Account information is contained in the user authorization database, which consists of the following files:

```
SYS$SYSTEM:SYSUAF.DAT          ! System User Authorization File (UAF)
                               ! (user account entries)

SYS$SYSTEM:RIGHTSLIST.DAT      ! Rights identifier database file
                               ! (all defined identifiers and holders)

SYS$SYSTEM:NETPROXY.DAT        ! Network proxy database file
                               ! (DECnet proxy entries)
```

The SYSUAF file contains most of the information usually associated with an account. The RIGHTSLIST file contains the definitions for all identifiers and tracks to whom they have been granted. Finally, the NETPROXY file contains proxy definitions used to determine access from remote DECnet users. This chapter focuses on managing the SYSUAF file. The procedures for managing identifiers and DECnet proxies are discussed in chapter 17.

As discussed in chapter 5, you have to coordinate access to the user authorization database files within a VMScluster environment. Usually, in a common-environment cluster, a single database is shared among cluster members and is located in a common directory: SYS$COMMON:[SYSEXE]. In a multiple-environment cluster, each cluster member typically maintains its own copy of database files, each located in a system-specific directory: SYS$SPECIFIC:[SYSEXE].

Regardless of your system configuration, system startup procedures should define logical names that designate the location for each database file. For example, to establish a shared database, define logical names as follows:

```
$ DEFINE/SYSTEM/EXEC SYSUAF        SYS$COMMON:[SYSEXE]SYSUAF.DAT
$ DEFINE/SYSTEM/EXEC RIGHTSLIST    SYS$COMMON:[SYSEXE]RIGHTSLIST.DAT
$ DEFINE/SYSTEM/EXEC NETPROXY      SYS$COMMON:[SYSEXE]NETPROXY.DAT
```

These commands should be included in SYLOGICALS.COM, as discussed in chapter 5.

A common authorization database greatly simplifies account administration. For example, if you have a six-member cluster that shares a single authorization database, you only have to create and maintain a single account for each user. On the other hand, without a shared database, a user who needs access to all six systems would need six separate accounts—one for each cluster member.

You manage entries in the SYSUAF file using the OpenVMS Authorize Utility, as follows:

```
$ RUN SYS$SYSTEM:AUTHORIZE
UAF>
```

By default, AUTHORIZE attempts to locate the authorization database by translating the logical names discussed above. If the logical names are not defined, AUTHORIZE searches your current default directory. You should define logical names for each database file so you won't have to manually set your default directory to SYS$SYSTEM before invoking AUTHORIZE.

You can further simplify access to AUTHORIZE by defining a symbol in your login command procedure, as follows:

```
$ UAF:==RUN SYS$SYSTEM:AUTHORIZE    ! Define symbol

$ UAF                               ! Invoke Authorize
UAF>
```

Refer to chapter 11 for more information on developing login command procedures.

9.4 Account Characteristics

Each user account has a number of characteristics that you configure depending on the user's requirements. The following output shows the characteristics for a sample account:

```
UAF> SHOW SIMPSONB

✚ Username: SIMPSONB     Owner: Barry Simpson
  Account: BILLING  UIC: [400,3] ([SIMPSONB])

✣ CLI: DCL     Tables: DCLTABLES
  Default: DSK$SYST:[SIMPSONB]
  LGICMD: LOGIN
  Flags: Restricted

★ Primary days: Mon Tue Wed Thu Fri
  Secondary days: Sat Sun
  Primary   000000000011111111112222    Secondary 000000000011111111112222
  Day Hours 012345678901234567890123    Day Hours 012345678901234567890123
  Network:  -------############-----              ------ No access -------
  Batch:    -------############-----              ------ No access -------
  Local:    -------############-----              ------ No access -------
  Dialup:   -------############-----              ------ No access -------
  Remote:   -------############-----              ------ No access -------
  Expiration: (none)          Pwdminimum: 6       Login Fails: 0
  Pwdlifetime: 30 00:00       Pwdchange: 19-APR-1995 15:32
  Last Login: (none) (interactive), (none) (noninteractive)

✱ Maxjobs:      0        Fillm:  100        Bytlm:   8192
  Maxacctjobs:  0        Shrfillm: 0        Pbytlm:     0
  Maxdetach:    0        BIOlm:   18        JTquota: 1024
  Prclm:        2        DIOlm:   18        WSdef:    150
  Prio:         4        ASTlm:   24        WSquo:    256
```

```
  Queprio:     0          TQElm:  10          WSextent:    512
  CPU:      (none)        Enqlm: 100          Pgflquo:   10240

✿ Authorized Privileges:
  NETMBX TMPMBX
  Default Privileges:
  NETMBX TMPMBX

♥ Identifier              Value               Attributes
  PRODX_READ              %X80010002
```

Account characteristics are grouped in the following six major categories:

- ✚ User identification
- ✚ Account defaults
- ★ Login attributes
- ✱ Quotas and limits
- ✿ Privileges
- ♥ Identifiers

Each of these categories is discussed in more detail in the following sections.

9.4.1 User Identification

Username

The account *username* is a string containing between 1 and 12 alphanumeric characters, including the dollar sign ($) and underscore (_) characters. The account username provides a mechanism for the system to identify a user in a textual way. Before users can log in, they must identify themselves by supplying their username in response to the *Username:* login prompt, as follows:

```
Username: SIMPSONB
```

Each user account must have a unique username. You derive usernames by combining characters from the user's first and last names. A common strategy is to truncate the user's last name to 11 (or fewer) characters and append the first letter of the user's first name. For example, Barry Simpson would be assigned the username SIMPSONB. If the resulting username is already in use, you can add more letters from the first name until you obtain a unique username, for example:

```
SIMPSONB, SIMPSONBA, SIMPSONBAR, etc.
```

If your site has a large number of users, you may have users with exactly the same first and last names. In that case, append characters from the user's middle name or append a sequential number.

Every account should be assigned to and used by *one person.* Avoid creating accounts like BILLING for use by several billing clerks. Individual accounts ensure that you can identify and locate a specific person in the event of a problem or a security breach.

User Identification Code (UIC)

Each OpenVMS file (and other system objects) has an associated owner. Digital could have used the account username to indicate ownership. That would have been inefficient, however, because a username is a relatively long character string. Instead, a Compact Numeric User Identification Code (UIC) is associated with each user account and is used to determine access to and ownership of system objects such as files, directories, devices, and queues. Refer to chapter 17 for more information on UIC-based access control.

A UIC consists of a *group* and a *member* number, separated by a comma, and enclosed in square brackets, as follows:

```
[group number, member number]
```

You specify group and member numbers using octal (base 8) limited to the following ranges:

Group: 1–37776
Member: 0–177776

Selecting appropriate UICs is one of the most important aspects of account administration. Even though the username and password control initial access, UICs determine which system objects users can access after they log in. Although it is not required, each user account should be assigned a unique UIC. If you assign the same UIC to multiple accounts, each is considered the same account in terms of access control.

Avoid using low UIC group numbers. UIC groups between 1 and the system parameter MAXSYSGROUP are granted system access. UIC groups 1 and 300 through 377 are reserved for use by Digital.

You should organize user accounts with similar access requirements into the same UIC group. By grouping users in this way you can use UIC-based

protection to establish access control, rather than having to implement a more complicated Access Control List (ACL)-based protection scheme (see chapter 17). For example, you might want to organize user accounts into the following UIC groups:

```
Account Name    UIC Group

EXECUTIV        [100,*]    ! Executives
BILLING         [200,*]    ! Billing Department
HR              [400,*]    ! Human Resources
SHIPPING        [500,*]    ! Shipping Department
SYSTEMS         [600,*]    ! Data Center Staff
```

When you add an account, specify a UIC using the following qualifier:

```
/UIC=[group, member]
```

UIC Identifiers

A UIC identifier is an alphanumeric synonym for the numeric UIC of an account. The identifier provides a more convenient way of specifying a UIC value. UIC identifiers are automatically created when you add an account. By default the identifier is the username for the account enclosed in square brackets. For example, the UIC identifier for SIMPSONB is listed in parentheses next to the numeric UIC for the account, as follows:

```
Username: SIMPSONB
...                        UIC: [100, 2] ([SIMPSONB])
```

When you add the first account in a particular UIC group, and you specify an account name (using the /ACCOUNT qualifier) a *group* identifier is also created. For example:

```
UAF> ADD SIMPSONM/ACCOUNT=HR/UIC=[400,1]
%UAF-I-RDBADDMSGU, identifier SIMPSONM value: [000400,0000001] added ...
%UAF-I-RDBADDMSGU, identifier HR value: [000400,177777] added ...
```

When there is a UIC group identifier, it is included in the UIC identifiers for all accounts in that same group. For example:

```
Username: SIMPSONM
...                        UIC: [400, 1] ([HR,SIMPSONM])
```

You can view the UIC identifier and UIC group identifiers for an account as follows:

```
UAF> SHOW/IDENTIFIER SIMPSONM

Identifier      Value           Attributes
SIMPSONM        [400,1]

UAF> SHOW/IDENTIFIER HR

Identifier      Value           Attributes
CHILD           [400,177777]
```

Note that the member UIC value of the identifier HR is greater than the maximum member number allowed for numeric UICs (177776). When an identifier has a member number of 177777 it is a group UIC identifier.

Passwords

Each account contains no passwords, a primary password, or both a primary and a secondary password. Each password can be a sequence of 0 to 31 alphanumeric characters, including the dollar sign ($) and underscore (_). You should encourage users to include numbers, the dollar-sign ($), and the underscore (_) in their passwords to make password guessing difficult.

Specify password options using the following authorize qualifiers:

```
/PASSWORD=(primary-pwd, secondary-pwd)
/NOPASSWORD
/PASSWORD=1IS_BETTER_THAN_NONE
/PASSWORD=(NOT_SO_EASY, EVEN_HARDER1$TO_GUESS)
```

Owner Name

An *owner* of an account indicates the person who is responsible for the account. Enter it as a character string containing 1 to 31 alphanumeric characters, as follows:

```
/OWNER="owner-name"
/OWNER="Barry Simpson"
```

If you have more than a few accounts, it is critical that you set the account owner to the full name of the user who is responsible for the account.

Account Name

Although misleading by its name, the *account* field is used to indicate the group or department with which a user is associated. It may contain 1 to 8 alphanumeric characters and is specified as follows:

```
/ACCOUNT="account-name"
/ACCOUNT="PAYROLL"
```

9.4.2 Account Defaults

Default Login Directory

The *default device* and *default directory* of an account specify the default directory selected after execution of system login procedures. The device name should always be expressed in a logical name, such as DISK$USER1. This allows you to move default directories from one physical disk to another without having to modify account information.

This information is configured using the following qualifiers:

```
/DEVICE=device-name
/DEVICE=DISK$USER1

/DIRECTORY=[dir-name]
/DIRECTORY=[SIMPSONB]
```

Default User Login Procedure

The *default login procedure* indicates the DCL command procedure to be executed after the system login procedure. It is specified using the following qualifiers:

```
/LGICMD=command-procedure
/LGICMD=LOGIN.COM
```

If the command procedure specified does not include a device and directory specification, this specification is taken from the default login device and directory for the account. If no login procedure is specified, LOGIN.COM is executed, if it exists. Login command procedures are discussed in more detail in chapter 11.

Command Line Interface (CLI)

The Command Line Interface (CLI) parameters select the default CLI the account will use. In most cases, select DCL as follows:

```
/CLI=DCL
/CLITABLES=DCLTABLES
```

9.4.3 Login Attributes

Each account has a number of characteristics that specify the following attributes.

Access Restrictions

The user authorization file (UAF) qualifiers listed in Table 9.1 determine when and from which login modes an account can access the system.

For example:

```
/ACCESS=(PRIMARY, 7-18)   ! Allow any access 7 A.M. to 6 P.M. (M-F)
/ACCESS=(7-19)            ! Allow any access 7 A.M. to 7 P.M. (M-S)
/NOBATCH=(7-18)           ! Prohibit batch access 7 A.M. to 6 P.M. (M-S)
```

Mail Attributes

```
/FLAG=DISMAIL        ! Prevents account from receiving OpenVMS mail
/FLAG=DISNEWMAIL     ! Suppresses new mail message count at login time
```

Login Message Restrictions

```
/FLAG=DISREPORT           ! Suppresses login time and failure messages
                          ! at login
/FLAG=DISWELCOME          ! Suppresses login welcome message at login
```

Password Attributes

```
/FLAG=NODISFORCE_PWD_CHANGE    ! Forces password change when expired
/PWDEXPIRED                    ! Pre-expires password
/FLAG=PWD_EXPIRED              ! Expire primary password
/FLAG=PWD2_EXPRIED             ! Expire secondary password
/FLAG=LOCKPWD                  ! Prevent user from changing password
/FLAG=GENPWD                   ! Force generated passwords
/PWDMINIMUM=min-length         ! Force password length >= min-length
/PWDLIFETIME=delta-time        ! Force regular password change
```

Other Security Attributes

```
/FLAG=AUDIT          ! Enables all auditing events for account
/FLAG=AUTOLOGIN      ! Restricts logins from Auto Login terminals (ALF)
/FLAG=DEFCLI         ! Forces use Default CLI and TABLES
```

Table 9.1 UAF Account Access Restrictions

Qualifier	*Controls*
/ACCESS=access-times	Access for all login modes
/BATCH=access-times	Access for batch logins
/DIALUP=access-times	Access from dial-up terminals
/INTERACTIVE=access-times	Access for all interactive logins
/LOCAL=access-times	Access for all local logins
/NETWORK=access-times	Access for network batch logins
/REMOTE=access-times	Access for remote logins (SET HOST)

```
/FLAG=DISUSER        ! Disables account
/FLAG=CAPTIVE        ! Configures accounts as Captive account
/FLAG=DISCTLY        ! Disables CTRL/Y interrupts during the login process
/FLAG=DISIMAGE       ! Prevents user from being able to issue RUN command
/FLAG=RESTRICTED     ! Configures accounts as a Restricted account
/FLAG=DISRECONNECT   ! Prohibits reconnect to disconnected process
```

Most of these characteristics are security related and are discussed in more detail in chapter 17.

9.4.4 Quotas and Limits

A number of account quotas and limits restrict the amount of system resources available to users. Refer to the documentation for each application you use at your site to determine the settings they require, as follows:

```
/ASTLM          ! Maximum outstanding Asynchronous System Traps
/BIOLM          ! Maximum outstanding buffered I/O requests
/BYTLM          ! Maximum byte count of outstanding buffered I/Os
/CPUTIME        ! Limit of CPU time used per session
/DIOLM          ! Maximum outstanding direct I/O requests
/ENQLM          ! Limits number of locks process can own
/FILLM          ! Limits number of simultaneous files process can open
/JTQUOTA        ! Bytes allocated for JOB logical names
/MAXACCTJOBS    ! Maximum interactive, batch, detached processes
/MAXDETACH      ! Maximum detached processes
/MAXJOBS        ! Maximum interactive, batch, detached processes,
                ! and network jobs (not counting first 4)
/PRCLM          ! Maximum number of subprocesses
/SHRFILLM       ! Maximum number of shared files that can be open
/TQELM          ! Limits outstanding timer queue entries
```

Scheduling priority controls how and when processes are scheduled to receive central processing unit (CPU) time. Accounts used for interactive purposes usually are assigned a base scheduling priority of 4

```
/PRIORITY       ! Default base scheduling priority
```

A number of account attributes control working set sizes, as follows:

```
/PGFLQUOTA      ! Page file quota in pages
/WSDEF          ! Working set default limit
/WSEXTENT       ! Working set extent limit
/WSQUOTA        ! Working set quota limit
```

Processes priority and working sets are discussed in more detail in chapter 18.

9.4.5 Privileges

Account privileges amplify user access and resource usage capabilities. Each account has associated *authorized privileges* and *default privileges*. A user may activate authorized privileges after login using the following command:

```
$ SET PROCESS/PRIVILEGE=(privilege,privilege,...)
```

When users log in, their associated default privileges are automatically activated.

Default and authorized privileges are configured using the following authorize qualifiers:

```
/DEFPRIVILEGE=(privilege,privilege,...)
/PRIVILEGE=(privilege,privilege,...)
```

Refer to chapter 17 for more information on OpenVMS privileges.

9.4.6 Identifiers

Identifiers are used in conjunction with access control lists (ACLs) to provide greater flexibility in determining access to system objects. Typically, a user account is granted an identifier that provides or restricts access to a file, queue, or other system object. Identifiers and ACLs are discussed in detail in chapter 17.

9.5 Creating Template Accounts

Digital supplies a pre-defined account named DEFAULT that determines the default account attributes for newly created accounts. When you add an account and do not specify a particular account attribute, it is assigned from the DEFAULT account. You should configure the DEFAULT account so that its attributes match those you most commonly use. This minimizes the number of account characteristics you must explicitly specify.

For example, if you decide that most of your accounts should have the following default attributes:

- Minimum password length of eight
- Password lifetime of 60 days
- Access time of Monday–Friday (8 A.M.–7 P.M), no weekend access

modify the DEFAULT account as follows:

```
UAF> MODIFY DEFAULT -
       /PWDLIFETIME="60-00:00"/PWDMINIMUM=8/ACCESS=(PRIMARY,8-19)
```

If you frequently add accounts, you should create more detailed template accounts for each major user group or application. For example

```
UAF> ADD TMPL_BILLING/ACCOUNT=BILLING/FILLM=30/PGFLQUO=25000 -
      /DEVICE=USR$BILLING/FLAG=DISUSER

UAF> ADD TMPL_HR/ACCOUNT=HR/WSDEF=1000/WSQUOTA=2000/WSEXTEN=5000 -
      /DEVICE=USR$HR/UIC=[400,1]/FLAG=DISUSER
```

9.6 Managing Accounts

9.6.1 Adding New Accounts

To add a new account, perform the following steps:

- Choose a username, UIC, initial password, and default directory
- Select the template account that most closely matches your needs
- Determine additional qualifiers not included in the selected template
- If you selected the DEFAULT template, use the authorize ADD command with any other required qualifiers

```
UAF> ADD SIMBSONB/OWNER="Barry Simpson"/UIC=[200,2] -
/ACCOUNT="KIDS"/PASSWORD=VERY2HARD/DIR=[SIMPSONB]/FLAG=NODISUSER
```

- If you selected a different template account, add the account by using the authorize COPY command

```
UAF> COPY DEF_BILLING SIMPSONM/OWNER="Mary Simpson" -
/UIC=[300,2]/PASSWORD=DIFFICULT9/DIRECTORY=[SIMPSONM] /FLAG=NODISUSER
```

- Add a disk quota entry if quotas are enabled on the user's default device (see chapter 10)
- Create the user's default directory

```
$ CREATE DISK$USER1:[SIMPSONB]/OWNER=[SIMPSONB]
```

- Create user login command procedure (see chapter 11)
- Add user to mail distribution lists (see chapter 16)

9.6.2 Disabling and Enabling Accounts

To temporarily enable and disable users accounts, issue the following commands:

```
UAF> MODIFY username/FLAG=NODISUSER
UAF> MODIFY username/FLAG=DISUSER
```

9.6.3 Deleting Accounts

To permanently remove a user account, perform the following steps:

- Archive user's files (if necessary)

```
$ BACKUP/VERIFY/DELETE DISK$USER1:[SIMPSONB...]*.*;* -
  ARCH$DIR:SIMBSONB.SAVE
```

Define the ARCH$DIR logical name to point to a directory that serves as a temporary holding area for archived accounts

- Remove disk quota entry (see chapter 10)

```
SYSMAN> DISKQUOTA REMOVE [SIMPSONB]/DEVICE=DISK$USER1
```

Table 9.2 Example Account Management Command Procedures

Command Procedure	*Function*
ACCT_ADDUSER.COM	Add a user account and home directory
ACCT_REMUSER.COM	Remove a user account and its home directory
ACCT_UAFRPT.COM	Produce a user account report

- Delete default directory and subdirectories

```
$ DELETE DISK$USER1:[SIMPSONB...]*.*;*
! Repeat above command until all subdirectories are deleted

$ DELETE DISK$USER1:[000000]SIMPSONB.DIR;  ! Delete home directory
```

- Remove user account using authorize REMOVE command

```
UAF> REMOVE SIMPSONB
```

- Remove user from mail distribution lists (see chapter 16)

9.6.4 User Account Reporting

The following command generates a report listing all user accounts:

```
UAF> LIST [*,*] /FULL     ! Detail report sorted by UIC
UAF> LIST [*,*] /BRIEF    ! Brief report by UIC

UAF> LIST * /FULL         ! Detail report sorted by Username
UAF> LIST * /BRIEF        ! Brief report by Username
```

By default, the report is written to SYS$SYSTEM:SYSUAF.LIS.

9.7 Tools

Table 9.2 lists several DCL command procedures that are located on the diskette included with this book (see Appendix E). These command procedures demonstrate techniques to automate the account management tasks discussed in this chapter.

9.8 Additional References

[OpenVMS 5.0]

VMS Authorize Utility Manual, Digital, April 1988.
Guide to Setting Up a VMS System, Digital, April 1988.

[OpenVMS 6.0]

OpenVMS System Manager's Manual: Essentials, chapter 6, "Managing User Accounts," Digital, March 1994.
OpenVMS System Management Utilities Reference, "AUTHORIZE" Utility, Digital, March 1994.

10 DISK SPACE MANAGEMENT

10.1 Chapter Objectives

- Prevent application downtime resulting from insufficient disk space
- Stabilize disk space use
- Encourage automatic disk space reclamation by OpenVMS and users

10.2 Introduction

Users depend heavily on the availability of disk space to create new files and extend existing ones. If a disk completely runs out of space, users' tasks fail, and the system itself may fail. To prevent application downtime, proper disk space management is essential.

An effective disk space management strategy should consider the following issues:

- Backup strategy
- Disk cluster size
- Purging files
- File version limits
- Disk quotas
- File expiration dates
- Volume rebuilds
- System log file archiving

10.3 Backup Strategy

A robust backup strategy is an important prerequisite to any disk space management strategy. If users do not feel comfortable with your backup strategy,

they may perform their own backups by storing duplicate versions of their files. Refer to chapter 13 for a detailed discussion of backup strategies.

10.4 Disk Cluster Size

When you initialize a disk for the first time, you establish its *cluster size* using the following command:

```
$ INITIALIZE/CLUSTER_SIZE=n disk-name disk-label
```

The cluster size of a disk specifies its minimum unit of allocation (in blocks). For example, if the cluster size of a disk is three, the size of the smallest file that can be created on that disk is three blocks. When the file is extended, it grows in increments of three blocks.

If you do not explicitly specify a cluster size when you initialize a disk, it defaults to the following:

3 for disks with >50,000 blocks
1 for disks with < 50,000 blocks

However, the minimum cluster size for very large disks is further limited to the result of the following formula:

$$\text{Minimum cluster size} = \frac{\text{Disk size (blocks)}}{255 * 4096}$$

The maximum cluster size you can use is one-hundredth the size of the disk.

A large cluster increases file performance, but at the expense of disk space. For example, a disk with a cluster size of three will have one allocated (but unused) block per file. Therefore, if you expect a disk to contain mostly small files, set its cluster size to the minimum allowed for the disk. On the other hand, if a disk has only a few large files, a larger cluster size helps performance.

10.5 OpenVMS Purge Utility

By default, when you create new versions of a file, prior file versions are *not* deleted. If you update a file frequently, you quickly accumulate many versions of the same file, consuming valuable disk space. For example, after changing the file LETTER.DOC ten times, the following versions exist:

```
$ DIR LETTER.DOC/DATE=MODIFIED/SIZE

Directory DISK$USER1:[SIMPSONB]

LETTER.DOC;10   100  28-FEB-1995 10:10
LETTER.DOC;9    100  28-FEB-1995 10:06
LETTER.DOC;8    100  28-FEB-1995 10:03
LETTER.DOC;7    100  28-FEB-1995 10:00
LETTER.DOC;6    100  28-FEB-1995 09:54
LETTER.DOC;5    100  28-FEB-1995 09:51
LETTER.DOC;4    100  28-FEB-1995 09:50
LETTER.DOC;3    100  28-FEB-1995 09:40
LETTER.DOC;2    100  28-FEB-1995 09:34
LETTER.DOC;1    100  28-FEB-1995 09:31

Total of 10 files, 1000 blocks.
```

To delete all but the most recent version, use the DCL PURGE command as follows:

```
$ PURGE LETTER.DOC

$ DIR LETTER.DOC/DATE=MODIFIED/SIZE

Directory DISK$USER1:[SIMPSONB]

LETTER.DOC;10   100  28-FEB-1995 10:10

Total of 1 file, 100 blocks.
```

If a disk contains many duplicate file versions that are not needed, purge the entire disk as follows:

```
$ PURGE DISK$USER1:[000000...]*.*
```

Caution: Use PURGE with extreme caution—remember you are *deleting* files. Some users or applications may need to maintain multiple file versions. To purge while retaining the last *n* versions of each file, perform the following command:

```
$ PURGE/KEEP=n DISK$USER1:[000000...]*.*
```

Extensive file purging has a severe effect on system performance. Identify disks or directories that require regular purging; assemble appropriate PURGE

commands into a DCL command procedure; and submit the procedure to run as a batch job during off-hours. Refer to chapter 12 for more information on creating automated batch jobs.

10.6 File Version Limits

Every file has a version limit attribute which specifies the maximum number of file versions that can exist at any one time. By default, files are assigned a version limit of zero—allowing an unlimited number of versions. Once you've establish version limits, you don't have to perform a PURGE because files are automatically purged when files reach their version limit.

File version limits are established as follows:

```
$ SET FILE/VERSION_LIMIT=2 LETTER.DOC;*

$ DIR LETTER.DOC/DATE=MODIFIED/SIZE

Directory DISK$USER1:[SIMPSONB]

LETTER.DOC;2    100  28-FEB-1995
LETTER.DOC;1    100  01-JAN-1995

Total of 2 files, 200 blocks.
```

If a new version of LETTER.DOC (version 3) is created, LETTER.DOC (version 1) is automatically deleted:

```
$ CREATE LETTER.DOC
TEST
^Z

$ DIR LETTER.DOC/DATE=MODIFIED/SIZE

Directory DISK$USER1:[SIMPSONB]

LETTER.DOC;3    100  15-MAR-1995
LETTER.DOC;2    100  28-FEB-1995

Total of 2 files, 200 blocks.
```

The amount of disk space used by the foregoing document remains a constant 200 blocks, thus stabilizing disk space usage.

To view a file's current version limit, issue the following command:

```
$ DIR/FULL file-name
$ DIR/FULL LETTER.DOC
Directory DISK$USER1:[SIMPSONB]

LETTER.DOC;2
.
.
File attributes:    Allocation: 6, Extend: 0, Global buffer count: 0
  Version limit: 2
.
.
```

10.6.1 Default File Version Limits

When you create files and directories, version limits are assigned as follows:

- When a new version of an *existing file* is created, its version limit is assigned from the previous version of the file
- When a *new file* is created, the version limit is set to the *default* version limit for that directory (specified by the parent directory)
- When a *new directory* is created, it is assigned the default version limit of its parent directory.

10.6.2 Default Directory Version Limit

Initially, directories are assigned a default version limit of zero, providing unlimited file versions. You can view a directory's default version limit by issuing the following command:

```
$ DUMP/HEAD/FORMAT/BLOCK=COUNT:0 DISK$USER1:[000000]SIMPSONB.DIR

Dump of file DISK$USER1:[000000]SIMPSONB.DIR;1 on 1-JUL-1995 09:52:30.85
File ID (1924,1,0) End of file block 1 / Allocated 2

Header Area                         File Header

Identification area offset:         40
Map area offset:                    100
Access control area offset:         255
Reserved area offset:               255
Extension segment number:           0
Structure level and version:        2, 1
```

```
File identification:                (1924,1,0)
Extension file identification:      (0,0,0)
  VAX-11 RMS attributes
Record type:                        Variable
File organization:                  Sequential
.
.
.
==> Directory version limit:        2
.
.
```

The version limit indicated when you issue a DIR/FULL command does *not* apply to the directory. You can view the directory version limit only by using DUMP/HEADER.

Modify the default version of a directory limit with this command:

```
$ SET DIRECTORY/VERSION_LIMIT=n [dir-name]
```

10.6.3 Establishing Version Limits

When you want to establish version limits for new directories, or directories that do not yet contain files, issue a SET DIRECTORY/VERSION_LIMIT command. To establish version limits on *existing* files you must perform a series of steps. For example, to establish a version limit of two on all files and subdirectories contained under DISK$USER1:[SIMPSONB], perform the following:

1. Ensure that users are not accessing files in the directories you want to manipulate. If users have open files, you will not be able to establish version limits
2. Set the *directory* version limit of the top-level directory and all subdirectories

   ```
   $ SET DIRECTORY/VERSION_LIMIT=2 DISK$USER1:[SIMPSONB...]
   ```

3. Set the *file* version limit on the *most recent* version of each file

   ```
   $ SET FILE/VERSION_LIMIT=2 DISK$USER1:[SIMPSONB...]*.*;
   ```

4. Use PURGE to delete extra file versions created before version limits were enabled

   ```
   $ PURGE DISK$USER1:[SIMPSONB...]*.*
   ```

10.7 Disk Quotas

Left unchecked, users can consume all available space on any disk to which they have write access. By establishing disk quotas, you can restrict space usage to a specific number of blocks per user. When they exceed their allocations, users cannot use additional space until they reduce their current usage below their quota.

Disk quotas encourage users to actively manage their own space usage. Every time they approach their quota, they are forced to immediately purge or delete files. Without quotas, *you* must become more involved in managing users' disk space.

10.7.1 Disk Quota Tools

Disk quotas are managed with the SYSMAN utility. Through OpenVMS version 5.5, you can also use the DISKQUOTA Utility. However, support for this utility will eventually be dropped so you should migrate to SYSMAN.

Disk quota commands issued from SYSMAN have the following format:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> DISKQUOTA disk-quota-command/qualifiers
```

10.7.2 Enabling Disk Quotas

Disk quotas are enabled and disabled on a disk by disk basis. To activate disk quotas, you must first create a *quota file*, as follows:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> DISKQUOTA CREATE/DEVICE=disk-name
```

When you enter the above command, the following occurs:

- The file disk-name:[000000]QUOTA.SYS is created. It contains disk quota entries and current disk usage statistics.
- The default quota entry ([0,0]) is added to the quota file. It determines default quotas assigned when not explicitly specified
- Quota entries are added for existing users. Any user who has existing files on the disk is given a default quota
- Disk quotas are enabled on the specified disk

If a disk contains a QUOTA.SYS file, disk quotas are automatically enabled when the disk is mounted. Therefore, exercise care when creating a QUOTA.SYS file. Even if you manually disable quotas, they will be automatically enabled when the disk is mounted during the next system boot.

To manually enable and disable disk quotas, issue this commands:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> DISKQUOTA ENABLE/DEVICE=disk-name
SYSMAN> DISKQUOTA DISABLE/DEVICE=disk-name
```

10.7.3 Rebuilding the Quota File

If disk quotas are disabled for a period of time, you must issue a REBUILD command to update the space usage statistics recorded in the quota file:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> DISKQUOTA REBUILD/DEVICE=disk-name
```

CAUTION: When rebuilding the quota file, users will not be able to allocate new space on the device. Therefore, you should only perform a rebuild when user activity is minimal. Also, be aware that disk quotas are automatically enabled after a rebuild.

10.7.4 Managing Quota Entries

After you enable disk quotas, you must manage quota entries for each user. Each entry has an associated *permanent* and an *overdraft* quota specified in blocks. Users can continue to allocate disk space as long as the total space used is less than their permanent quota. Users may exceed their permanent quota to append to an existing file until the additional space used exceeds their overdraft limit.

The following illustrates disk quota enforcement:

```
$ SHOW QUOTA

User [SIMPSONB] has 6435 blocks used, 5 available, of 6440 authorized
and permitted overdraft of 10000 blocks on DISK$USER1
```

This user has only five blocks of permanent disk quota available. If the user attempts to create a new file larger than five blocks, the user's disk quota will be exceeded, and the operation is not allowed to complete:

```
$ DIRECTORY/SIZE=ALL T.TMP

Directory DISK$USER1:[SIMPSONB]

T.TMP;2                    17/18

Total of 1 file, 17/18 blocks.

$ COPY T.TMP T.TMP

%COPY-E-OPENOUT, error opening DISK$USER1:[SIMPSONB]T.TMP; as output
-RMS-E-CRE, ACP file create failed
-SYSTEM-F-EXDISKQUOTA, disk quota exceeded

%COPY-W-NOTCOPIED, DISK$USER1:[SIMPSONB]T.TMP;2 not copied
```

However, consider the following user who attempts to edit and save a file when he is nearly out of disk quota:

```
$ SHOW QUOTA

User [SIMPSONB] has 6435 blocks used, 5 available, of 6440 authorized
and permitted overdraft of 10000 blocks on DISK$USER1

$ EDIT T.TMP
    1 ************************************************************
*EXIT

DISK$USER1:[SIMPSONB]T.TMP;3 100 lines

$ SHOW QUOTA
User [SIMPSONB] has 6454 blocks used, 14 OVERDRAWN, of 6440
authorized and permitted overdraft of 10000 blocks on $DISK1
```

This operation was successful because the OpenVMS Editor initially allocates a small file and then appends to it during the EXIT operation. Because the user had available overdraft quota, the operation was allowed to complete.

10.7.4.1 Adding Quota Entries

Add disk quota entries using the following command:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> DISKQUOTA ADD [uic-value]/PERMANENT=blocks /OVERDRAFT=blocks -
  /DEVICE=disk-name
```

The uic-value you specify should be either the User Identification Code (UIC) or UIC identifier (as listed in the User Authorization File [UAF]). For example, if a user has the following account information:

```
Username:           SIMPSONB
UIC:                [150,1]
UIC Identifier:     [SIMPSONB]
```

you may add a quota entry by specifying a uic-value of either [150,1] or [SIMPSONB]. You can also specify SIMPSONB. However, remember that this works because SIMPSONB is an abbreviated from of the full UIC identifier. Disk quota entries are managed by UIC, *not* by user name.

10.7.4.2 Listing Quota Entries

To list an existing quota entry use the following command:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> DISKQUOTA SHOW [SIMPSONB] /DEVICE=disk-name
```

To list all quota entries use the following command:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> DISKQUOTA SHOW [*,*] /DEVICE=disk-name
```

For example:

```
SYSMAN> DISKQUOTA SHOW [*,*] /DEVICE=DISK$USER1

%SYSMAN-I-QUOTA, disk quota statistics on device DISK$USER1 - NODE BIRCH

UIC             Used        Permanent Quota      Overdraft Limit
[0,0]           0           1000                 100
[SIMPSONB]      6454        6440                 10000
.
.
.
```

10.7.4.3 Removing Quota Entries

To remove a disk quota entry:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> DISKQUOTA REMOVE [uic-value]/DEVICE=disk-name
```

10.7.5 Soft Disk Quotas

Enforced disk quotas are not appropriate for every situation. If users are unsophisticated or you can't afford to risk that an application will exhaust disk quota, you may want to consider using soft disk quotas.

Soft disk quotas are established by means of the same procedures discussed in the previous sections, except each user is given a permanent quota equivalent to the disk size. This way, users can never exceed their quotas. Although soft quotas do not directly limit disk space usage, the advantage is that now you can use the SYSMAN utility as a monitoring tool. For example, by periodically generating usage reports you can track overall usage trends and identify which users are consuming space at the highest rate.

To generate a usage report for a particular device, use the following commands:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> DISKQUOTA SHOW [*,*]/DEVICE=disk-name
```

Frequently, users don't bother deleting files simply because they don't realize just how much space they're using. With soft quotas enabled you can add commands to system login procedures to let users know just how much they are using. For example:

```
$! SYLOGIN.COM
$!
.
.
$ SHOW QUOTA

User [SIMPSONB] has 6443 blocks used, 93557 available, of 100000
authorized and permitted overdraft of 10000 blocks on DISK$USER1
```

10.8 Using File Expiration Dates

Each file has an associated *file expiration date* that can be used to determine when a file can be deleted or archived. By default, the file expiration date of each file is set to "none," indicating that the file has no expiration date. To establish expiration dates, issue the following command:

```
$ SET FILE/EXPIRATION=expiration-date file-name
```

A file is considered *expired* when its expiration date is before the current date. OpenVMS does *not* initiate any action when files become expired (e.g., the file is *not* deleted). Expired files can, however, be manipulated using date selection qualifiers of utilities such as COPY, DELETE, DIRECTORY, and BACKUP. For example, to archive all expired files to tape, issue the following command:

```
$ BACKUP/EXPIRED/SINCE=TODAY/DELETE DISK$USER1:[000000...]*.*;* MUA0:ARCH.SAV
```

CAUTION: Files that have no expiration date are considered expired (because "none" is always considered to be before the current date).

To automatically establish expiration dates for all files on a specific disk, you must enable *volume retention*. Enable volume retention by specifying a minimum and maximum retention time using the following command:

```
$ SET VOLUME/RETENTION=(minimum-time, maximum-time) disk-name
```

The retention times must be expressed in the OpenVMS delta time format. For example, to establish a minimum retention time of 30 days and a maximum retention time of 60 days, issue the following command:

```
$ SET VOLUME/RETENTION=("30-","60-") DISK$USER1
```

When volume retention is enabled on a disk, newly *created* files are assigned an expiration date equal to the sum of the current date and the *maximum* retention time. Then, each time a file is *accessed*, the sum of the current date and the *minimum* retention time is calculated. If this sum is a date later than the current expiration date of the file, a new expiration date is set to the sum of the current date plus the *maximum* retention time.

With a minimum retention time of 30 days and a maximum of 60 days, a file when it is first created or accessed is assigned an expiration date of the current time plus 60 days. Then any accesses over the next 30 days would *not* cause the expiration date to be updated. If the file is accessed after the thirtieth day, its expiration date would again be set 60 days in the future. By selecting a minimum retention time that is sufficiently large, you minimize the amount of system overhead required to maintain expiration dates.

For the purposes of volume retention, only accesses with EDIT, TYPE, OPEN, or COPY constitute a file access. Although BACKUP and DIRECTORY commands access a file, they do *not* cause file expiration dates to be modified. For example, when using BACKUP and DIRECTORY:

```
$ DIR/DATE=(MODIFIED,EXPIRED) T2.TMP

Directory DISK$USER1:[SIMPSONB]

T2.TMP;1        18-FEB-1995 13:09:14.08 19-MAY-1995 14:00:17.53

Total of 1 file.

$ BACKUP T2.TMP T3.TMP

$ DIR/DATE=(MODIFIED,EXPIRED) T2.TMP

Directory DISK$USER1:[SIMPSONB]

T2.TMP;1        18-FEB-1995 13:09:14.08 19-MAY-1995 14:00:17.53

Total of 1 file.
```

The expiration date of the file does *not* change. However, when COPY is used, the date does change:

```
$ SHOW TIME
  20-APR-1995 15:21:29

$ COPY T2.TMP T3.TMP

$ DIR/DATE=(MODIFIED,EXPIRED) T2.TMP

Directory DISK$USER1:[SIMPSONB]

T2.TMP;1        18-FEB-1995 13:09:14.08 19-JUN-1995 15:21:46.21

Total of 1 file.
```

WARNING: Because volume retention requires several calculations and possibly a file update every time a file is accessed, performance may degrade on heavily utilized disks.

Enabling volume retention provides a very powerful tool for managing disk space. When files expire, you know that they have not been accessed in at least 30 days (or the minimum retention time you specify). Once you identify files that have not been accessed recently, you can ask users whether you can archive or delete their expired files. To identify the amount of disk space used by expired files, issue the following command:

```
$ DIR/EXPIRED/BEFORE=TODAY/SIZE=ALL disk-name:[000000...]
```

10.8.1 Establishing Volume Retention

When you enable volume retention, expiration dates are set only when you create or access a file. For example, after you enable volume retention on DISK$USER1 with the following command:

```
$ SET VOLUME/RETENTION=("30-","60-") DISK$USER1
```

expiration dates are not automatically set, as can be seen in the following:

```
$ DIR T.TMP/DATE=EXP

Directory DISK$USER1:[SIMPSONB]

T.TMP;6 <None specified>
T.TMP;5 <None specified>

Total of 2 files.
```

You must first access the file, as follows:

```
$ SH TIME

  18-FEB-1995 13:05:34

$ TYPE T.TMP;*
$ DIR/DATE=(MODIFIED,EXPIRED) T.TMP

Directory DISK$USER1:[SIMPSONB]

T.TMP;6    18-FEB-1995 13:06:40.93   18-APR-1995 13:06:40.93
T.TMP;5    18-FEB-1995 13:06:41.05   18-APR-1995 13:06:41.05

Total of 2 files.
```

Note that the file expiration date is set to the current time plus the *maximum* retention time of 60 days. When the file is accessed 1 month (30 days) later, its expiration date is again set to the current time plus the *maximum* retention time of 60 days:

```
$ SH TIME
  20-MAR-1995 14:00:01

$ TYPE T2.TMP
ASDF
```

```
$ DIR/DATE=(MODIFIED,EXPIRED) T2.TMP
Directory $DISK1:[USER.BALDWIN]

T2.TMP;1        18-FEB-1995 13:09:14.08 19-MAY-1995 14:00:17.53
```

To force expiration dates to be updated on an entire volume, copy the entire disk to the NULL device when no files are open on the disk, as follows:

```
$ COPY DISK$USER1:[000000...]*.*;* NL:
```

10.8.2 Other Selection Commands

The following are additional commands for manipulating expired files. Substitute DELETE or BACKUP for DIR to perform other functions:

```
$ DIR/BEFORE=TODAY/EXPIRED     ! list all expired files
$ DIR/SINCE="-30-"/EXPIRED     ! files that expired in the last 30 days
$ DIR/SINCE="30-"/BEFORE="59-"/EXPIRED  ! all files accessed today
```

10.9 Volume Rebuild

Periodically the amount of disk space reported by SHOW DEVICE does not indicate the correct amount of space in use. This problem occurs because space allocation information is stored in memory and is not written to disk when disks are dismounted improperly (e.g., after a system crash).

For example:

```
$ SHOW DEVICE DISK$USER1

Device     Device     Error     Volume      Free     Trans      Mnt
Name       Status     Count      Label     Blocks    Count      Cnt
DUA0:      Mounted        0      USER1     17324      118        1
```

To correct the problem issue the following command:

```
$ SET VOLUME/REBUILD DISK$USER1
$ SHOW DEVICE DISK$USER1

Device     Device     Error     Volume      Free     Trans      Mnt
Name       Status     Count      Label     Blocks    Count      Cnt
DUA0:      Mounted        0      USER1     18356      118        1
```

By default, disks are mounted with the /REBUILD option, which performs the same function as SET VOLUME/REBUILD. If you mount disks at system

startup time with the /REBUILD option, you do not need to issue a manual rebuild. However, because volume rebuilds can take a long time, many system managers mount disks with the /NOREBUILD option so that systems can recover from a crash much faster. If you use this technique (see STARTUP_MOUNT in chapter 5), you should make sure you manually rebuild each disk after a system crash using SET VOLUME/REBUILD.

CAUTION: Exercise care in performing volume rebuilds when users are accessing the system. Rebuilds can degrade performance on the volume being rebuilt.

10.10 Archiving System Log Files

OpenVMS maintains the following log and data files, which can consume a great deal of disk space if not archived or deleted on a regular basis:

- ACCOUNTNG.DAT — Resource accounting data
- SECURITY_AUDIT.AUDIT$JOURNAL — Security audit database
- OPERATOR.LOG — Operator log file
- ERRORLOG.SYS — Device error log file

The following procedures copy the current system files to an archive area TASM$ARCHIVE. Once all files are moved to the archive area you can use the backup utility to archive files to tape.

Accounting Data File: ACCOUNTNG.DAT

```
$ SET ACCOUNTING/NEW_FILE
$ BACKUP/DELETE SYS$SPECIFIC:[SYSMGR]ACCOUNTNG.DAT;-1 -
      TASM$ARCHIVE:ACC_mmm_yy.DAT
```

Security Audit Journal File: AUDIT$SECURITY.JOURNAL

```
$ SET AUDIT/LOG=NEW_FILE
SYS$COMMON:[SYSMGR]SECURITY_AUDIT.AUDIT$JOURNAL
$ BACKUP/DELETE SYS$COMMON:[SYSMGR]SECURITY_AUDIT.AUDIT$JOURNAL;-1 -
      TASM$ARCHIVE:SEC_mmm_yy.DAT
```

System Error Log File: ERRORLOG.SYS

```
$ BACKUP/DELETE SYS$SPECIFIC:[SYSERR]ERRORLOG.SYS; -
      TASM$ARCHIVE:ERR_mmm_yy.DAT
```

Operator Log File: OPERATOR.LOG

```
$ REPLY/ENABLE=ALL
$ REPLY/LOG
$ BACKUP/DELETE SYS$COMMON:[SYSMGR]OPERATOR.LOG;-1 -
      TASM$ARCHIVE:OPER_mmm_yy.DAT
```

10.11 Tools

CUTTER

A DCL command procedure, called CUTTER, is located on the diskette included with this book (see Appendix E). It automatically archives system log files, as discussed in the previous sections.

CHECK_DISKS

No matter how robust your space management strategy, some user or application will still unintentionally consume all disk space. Therefore, in addition to implementing the strategies discussed in this chapter, you should also deploy an automated monitoring system that notifies you when space gets low. Such a tool, CHECK_DISKS, is presented in chapter 15.

10.12 Additional References

[OpenVMS 5.0]

SYSMAN Utility Manual, Digital, June 1990.

[OpenVMS 6.0]

OpenVMS DCL Dictionary, "DCL INITIALIZE Command," Digital, March 1994.

OpenVMS System Manager's Manual: Essentials, chapter 8, "Managing Storage Media," Digital, March 1994.

11 LOGIN COMMAND PROCEDURES

11.1 Chapter Objectives

- Create organized login procedures
- Minimize login time
- Develop a simple menu system

11.2 Introduction

Login command procedures are DCL command files that are executed during the login process. The purpose of these procedures is to customize a user's environment to facilitate access to applications. Not all users have the same requirements, so you may need to establish several levels of login procedures. For example, you may need to establish a system login procedure that everyone executes; a group login procedure for each major user group; and a user login procedure for each user. If you maintain a VMScluster, you may need to establish system login procedures for each cluster member.

Normally, after login procedures execute, users receive the DCL prompt and can enter DCL commands until they logout. If you have users who are unfamiliar with DCL, you can establish a login procedure that provides a menu interface to DCL. You may want to restrict access to DCL altogether by creating a *captive* login procedure.

Login procedures usually define symbols, logical names, or function keys, which allow users to access their applications. When login procedures fail, users may not be able to access their applications. Therefore, you must be extremely careful whenever you modify login procedures.

11.3 Login Sequence

Login procedures are executed immediately after a user enters a valid username and password combination, as follows:

1. User enters account and password.

```
Username: SIMPSONB <RETURN>
Password: dont_have_a_cow <RETURN>
```

2. System login procedure executed. Defined by logical name SYS$SYLOGIN (SYS$MANAGER:SYLOGIN.COM, by default).

3. User login procedure executed. Defined by User Authorization File (UAF) (see chapter 9) parameters on a per-user basis.

```
/LGICMD=default-user-login-procedure
/DEVICE=default-login-device
/DIRECTORY=default-login-directory
```

   Usually, LOGIN.COM is located in user's default directory.

4. User receives DCL prompt.

```
$
```

The following sections discuss system and user login procedures in more detail.

11.4 System Login Command Procedure

The system login procedure is located by the system logical name SYS$SYLOGIN, which is typically defined in system startup procedures (e.g., SYLOGICALS.COM) as follows:

```
$ DEFINE/SYSTEM/EXEC SYS$SYLOGIN SYS$COMMON:[SYSMGR]SYLOGIN.COM
```

The purpose of SYLOGIN is to define frequently used symbols and to define logical names that must be defined at the process level. System-wide logical names should be defined during system startup (see chapter 5).

Typically, SYLOGIN performs the following functions:

- Disable interrupts
- Execute mode-specific login procedures
- Configure terminal type
- Set the DCL Prompt
- Define function keys
- Define symbols
- Assign default system printer
- Set default file protection
- Execute group login procedures
- Execute cluster login procedures

11.4.1 Disable Interrupts

To prevent users from aborting the execution of system login procedures, you should temporarily disable CTRL/Y interrupts , as follows:

```
$!
$! SYLOGIN.COM
$!
$! Start of procedure
$ SET NOCONTROL=Y          ! Disable Interrupts
$ ...Body of procedure
$! End of procedure
$ SET CONTROL=Y            ! Enable Interrupts
```

However, even if you disable interrupts in this way, users can still abort the system login procedure by pressing CTRL/Y immediately after entering their password. To more effectively disable interrupts, set the UAF RESTRICTED flag on each user account, as follows:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE

UAF> MODIFY username /FLAGS=RESTRICTED
```

The RESTRICTED flag disables CTRL/Y interrupts from the moment a login is initiated, through the completion of both system and user login procedures (see chapter 9 for more information).

11.4.2 Execute Mode-specific Login Procedures

The system login procedure is executed not only for interactive logins but also for network and batch logins. Network and batch jobs do not have an associated terminal; therefore, you must design login procedures so that non-interactive jobs bypass interactive-specific commands. For example, a batch job shouldn't execute a SET TERMINAL/INQURE.

To detect this, use the F$MODE lexical function to skip over inappropriate parts of your login procedure, as follows:

```
$ IF F$MODE() .EQS. "INTERACTIVE" THEN GOTO INTERACTIVE
$ IF F$MODE() .EQS. "BATCH" THEN GOTO BATCH
$ IF F$MODE() .EQS. "NETWORK" THEN GOTO COMMON
$ GOTO COMMON             ! All other modes.
$!
$INTERACTIVE:
$! ... commands for interactive mode only
$ GOTO COMMON
$!
$BATCH:
$! ... commands for batch mode only
$ GOTO COMMON
$!
$COMMON:
$! ... commands for all modes
$ EXIT
```

11.4.3 Set Terminal Type

OpenVMS and most applications format output to terminals differently depending on terminal type. During system startup (see chapter 5), the device characteristics for physically attached terminal ports (e.g., TXA0: and TTA0:) are configured in system startup procedures (e.g., STARTUP_SETTERM.COM), as follows:

```
$ SET TERMINAL/DEVICE=VT300 TXA0:
$ SET TERMINAL/DEVICE=VT400 TXA1:
etc.
```

However, when users log in from a terminal server or modem you never know what type of terminal is being used until after they log in. Thus, in these cases, terminal characteristics must be set during Login. This is accomplished using the following command:

```
$ SET TERMINAL/INQUIRE
```

When you issue the above command, OpenVMS queries the terminal to send its device type. The terminal responds by sending back a special sequence of characters, then OpenVMS issues the equivalent of a SET TERMINAL/DEVICE=terminal-type. The most important disadvantage of this inquiry process is that it can take as long as 5 seconds to complete. If minimizing login time is important, you should design your login procedures so that SET TERMINAL/INQUIRE is executed only when absolutely necessary.

11.4.4 Set DCL Prompt

The standard DCL prompt is the dollar-sign ($). If you have multiple systems in a VMScluster, you may want to modify the default prompt to include the system node name as follows:

```
$ SET PROMPT="''f$GETSYI("NODENAME")'$ "
BIRCH$
```

This is especially useful when managing multiple systems. When you switch back and forth between systems it's easy to forget which system you're logged into.

11.4.5 Define Function Keys

To make entering commands even easier, define function keys for the most commonly used commands as follows:

```
$ DEFINE/KEY PF1 "DIRECTORY"/TERMINATE/ECHO
$ DEFINE/KEY PF2 "SET DEFAULT"/TERMINATE/ECHO
```

11.4.6 Define Global Symbols

If users execute certain DCL commands frequently, you can create symbols to make it easier for them. For example

```
$ DIR      :==DIRECTORY/DATE/SIZE=ALL
$ LO*GOUT  :==LOGOUT/FULL
$ PRINT    :==PRINT/NOTIFY
$ HOME     :==SET DEFAULT SYS$LOGIN
$ SD       :==SHOW DEFAULT
$ DEL*ETE  :==DELETE/CONFIRM
```

11.4.7 Assign Default System Printer

As discussed in chapter 8, when users send print jobs without explicitly specifying a print queue, the jobs are sent to the queue defined by the logical name SYS$PRINT. System startup procedures define the *system* logical name SYS$PRINT that defines the default system printer. However, during the login process you can define a SYS$PRINT job logical to define a default print queue for each specific user.

In general, there is a user characteristic that you capture during the login process from which you can determine a default print queue. For example, to extract this information, add the following commands to your login procedure:

```
$ uic_group =  f$getjpi("","GRP")              ! UIC group
$ account =    f$getjpi("","ACCOUNT")          ! UAF Account name
$ terminal =   f$getjpi("","TERMINAL")         ! Terminal name
$ term_serv =  f$getdvi(terminal,"TT_ACCPORNAM")   ! Term. Svr name
$!
$! Establish Default printer
$!
$ if uic_group .EQ. 150 then DEFINE/JOB SYS$PRINT SALES_LN03
$ if uic_group .EQ. 160 then DEFINE/JOB SYS$PRINT HR_LN03
$ if account .EQ. "SHIPPING" then DEFINE/JOB SYS$PRINT SHIPPING_LP37
etc...
```

For example, if all users in the Sales Department have the same UIC group number ([150,*]), you could automatically assign their default print queue as SALES_LN03.

11.4.8 Set Default File Protection

OpenVMS establishes a default protection mask during login based on the value of the system parameter RMS_FILEPROT. By default, RMS_FILEPROT specifies the following protection mask:

```
(S:RWED, O:RWED, G:RE, W:)
```

When users create *new* files, the file is assigned the user's default protection mask. If the standard default mask specified by RMS_FILEPROT is not acceptable for your site, you should either modify RMS_FILEPROT (see chapter 17) or add a SET PROTECTION command to login procedures. For example, to remove group read and execute protection, add the following command:

```
$ SET PROTECTION=(S:RWED,O:RWED,G,W)/DEFAULT
```

11.4.9 Group Login Procedures

Establish group login procedures to configure symbols or logical names specific to a particular set of users. For example, you could create a login procedure for the sales, human resources, and shipping groups, as follows:

```
LOGIN_SALES.COM
LOGIN_HR.COM
LOGIN_SHIPPING.COM
```

Then add conditional statements to the system login procedure to execute the appropriate group login procedure, as follows:

```
$! SYLOGIN.COM
$! ...
$ uic_group = f$getjpi("","GRP")
$ if uic_group .EQ. 150 then @SYS$MANAGER:LOGIN_SALES
$ if uic_group .EQ. 160 then @SYS$MANAGER:LOGIN_HR
$ if uic_group .EQ. 170 then @SYS$MANAGER:LOGIN_SHIPPING
```

11.4.10 Cluster Login Procedures

If your systems are stand-alone or members of a common-environment cluster, you can maintain a single, common set of system login command procedures. On the other hand, if you maintain a multiple-environment cluster, or if there are some differences in the applications available throughout the cluster, you may need a common login procedure and one or more system-specific login procedures.

This can be accomplished by using the same basic strategy as for system startup procedures (see chapter 5). If there are not many differences between cluster members, you may want to maintain a common login procedure but surround member-specific commands with IF-THEN statements. For example:

```
$! SYLOGIN.COM
$!
$ NODE = F$GETSYI("NODENAME")
$ ...
$ ... common login procedures
$ ...
$ IF NODE .EQS. "BIRCH"
```

```
$ THEN
$ ... BIRCH-specific login procedures
$ ENDIF
$ IF NODE .EQS. "LOTUS"
$ THEN
$ ... LOTUS-specific login procedures
$ ENDIF
etc...
```

On the other hand, if there are many differences between cluster members, you may want to have a common system login procedure that invokes a member-specific login procedure for each system. For example:

```
$! SYLOGIN.COM
$!
$ NODE = F$GETSYI("NODENAME")
$ ...
$ ... common login procedures
$ ...
$ @SYS$MANAGER:SYLOGIN_'NODE'.COM   ! Execute system-specific procedures

$!
$! SYLOGIN_BIRCH.COM
$ ...
$ ... BIRCH-specific login procedures
$ ...

$!
$! SYLOGIN_LOTUS.COM
$ ...
$ ... LOTUS-specific login procedures
$ ...
```

11.4.11 Template System Login Procedures

The following is an example SYLOGIN.COM that implements the techniques discussed in the previous sections:

```
$! SYLOGIN.COM       - Common login procedure
$!
$ SET NOCONTROL=Y              ! Disable interrupts
$!
$ NODE = F$GETSYI("NODENAME")
$ IF F$MODE() .EQS. "INTERACTIVE" THEN GOTO INTERACTIVE
$ IF F$MODE() .EQS. "BATCH" THEN GOTO BATCH
$ IF F$MODE() .EQS. "NETWORK" THEN GOTO COMMON
```

```
$ GOTO COMMON              ! All other modes
$! ------------------------------------------------------
$! Perform login commands appropriate for Interactive only
$! ------------------------------------------------------
$INTERACTIVE:
$ terminal = f$getjpi("","TERMINAL")                ! String
$ term_serv = f$getdvi(terminal,"TT_ACCPORNAM")     ! String
$!
$! Set terminal type (unless TT or TX port)
$!
$ term_type = f$extract(0,2,f$getjpi("","TERMINAL"))
$ dialup = f$getdvi(terminal,"TT_DIALUP")
$ if dialup
$ then
$  SET TERMINAL/INQUIRE
$ else
$  if (term_type .NES. "TX" .and. term_type .NES. "TT") then -
   SET TERMINAL/INQUIRE
$ endif
$!
$! Set DCL prompt
$!
$ SET PROMPT="''NODE'$ "
$!
$! Define Function Keys
$!
$ DEFINE/KEY PF1 "DIRECTORY"/TERMINATE/ECHO
$ DEFINE/KEY PF2 "SET DEFAULT"/TERMINATE/ECHO
$ GOTO COMMON
$! ------------------------------------------------
$! Perform login commands appropriate for BATCH only
$! ------------------------------------------------
$BATCH:
$!
$! ------------------------------------------------
$! Perform login commands appropriate all login modes
$! ------------------------------------------------
$COMMON:
$ uic_group =   f$getjpi("","GRP")              ! Integer
$ account =     f$getjpi("","ACCOUNT")          ! String
$!
$! Define system-wide symbols
$!
$ DIR      :==DIRECTORY/DATE/SIZE=ALL
$ LO*GOUT  :==LOGOUT/FULL
$ PRINT    :==PRINT/NOTIFY
$ HOME     :==SET DEFAULT SYS$LOGIN
```

```
$ SD       :==SHOW DEFAULT
$ DEL*ETE :==DELETE/CONFIRM
$!
$! Set default file protection
$!
$ SET PROTECTION=(S:RWED,O:RWED,G,W)/DEFAULT
$!
$! Execute Group Login procedure
$!
$ uic_group = f$getjpi("","GRP")
$ if uic_group .EQ. 150 then @SYS$MANAGER:LOGIN_SALES
$ if uic_group .EQ. 160 then @SYS$MANAGER:LOGIN_HR
$ if uic_group .EQ. 170 then @SYS$MANAGER:LOGIN_SHIPPING
$!
$ SET CONTROL=Y           ! Enable interrupts
$!
$! Execute system-specific login procedures
$!
$ @SYS$MANAGER:SYLOGIN_'NODE'.COM
$ exit
```

11.5 User Login Command Procedure

After the system login procedure is executed, OpenVMS determines if a *user login* procedure should be executed. To determine the name of this file, OpenVMS checks the user's UAF record. If the UAF LGICMD attribute contains a complete file specification that includes device, directory, and filename, that command procedure is executed. However, LGICMD normally contains only a filename, such as LOGIN. In this case, the user's default login device and directory are used to generate the complete file specification.

For example, if a user account has the following UAF parameters:

```
LGICMD:        LOGIN
DEVICE:        DISK$USER1
DIRECTORY:     [SMITH]
```

the default user login procedure would be

```
DISK$USER1:[SMITH]LOGIN.COM
```

As long as you design your system login procedures correctly, most users should need very few commands in their login procedure. However, more experienced users may add commands to define personal symbols and logical names to suit their needs.

11.6 Reducing Login Time

If users experience login times that are unacceptable, you should begin by identifying exactly were most of the time is being spent. You should first identify if most of time is spent in system login procedures (e.g., SYLOGIN.COM) or in user login procedures (e.g., LOGIN.COM). Start by timing the entire login process for a given user. Next, temporarily rename the user's login procedure to prevent it from executing, then time the login process again. If login time decreases considerably, you know to focus your efforts on the user login procedure; otherwise focus on the system login procedure.

Once you isolate the problem to either the user or system procedure, the next step is to identify exactly which statements within the procedures are taking the most time. If you are running OpenVMS Versions 5.5 or higher you can use the SET PREFIX command to enable checkpointing as follows:

```
$ SET PREFIX "<!12%T>"
$ SET VERIFY
$ @SYLOGIN/OUTPUT=SYLOGIN.LOG
$ TYPE SYLOGIN.LOG

<16:33:22.11 >$! SYLOGIN.COM - Common login procedure
<16:33:22.11 >$!
<16:33:22.12 >$ SET NOCONTROL=Y          ! Disable interrupts
<16:33:22.13 >$!
<16:33:22.14 >$ NODE = F$GETSYI("NODENAME")
<16:33:22.15 >$ IF F$MODE() .EQS. "INTERACTIVE" THEN GOTO INTERACTIVE
<16:33:22.20 >$INTERACTIVE:
<16:33:22.20 >$ terminal = f$getjpi("","TERMINAL")          ! String
<16:33:22.22 >$ term_serv = f$getdvi(terminal,"TT_ACCPORNAM") ! String
<16:33:22.25 >$!
<16:33:22.26 >$! Set terminal type (unless TT or TX port)
<16:33:22.26 >$!
<16:33:22.27 >$ term_type = f$extract(0,2,f$getjpi("","TERMINAL"))
<16:33:22.30 >$ dialup = f$getdvi(terminal,"TT_DIALUP")
<16:33:22.32 >$ if dialup
<16:33:22.34 >$ else
<16:33:22.34 >$ if (term_type .NES. "TX" .and. term_type .NES. "TT") -
   then SET TERMINAL/INQUIRE
<16:33:22.93 >$ endif
<16:33:22.95 >$!
<16:33:22.95 >$! Set DCL prompt
<16:33:22.95 >$!
<16:33:22.96 >$ SET PROMPT="BIRCH$ "
<16:33:22.98 >$!
<16:33:22.98 >$! Define Function Keys
```

```
<16:33:22.98 >$!
<16:33:22.99 >$ DEFINE/KEY PF1 "DIRECTORY"/TERMINATE/ECHO
<16:33:23.08 >$ DEFINE/KEY PF2 "SET DEFAULT"/TERMINATE/ECHO
<16:33:23.18 >$ GOTO COMMON
<16:33:23.20 >$COMMON:
<16:33:23.21 >$ uic_group = f$getjpi("","GRP")      ! Integer
<16:33:23.23 >$ account = f$getjpi("","ACCOUNT")    ! String
<16:33:23.25 >$!
<16:33:23.25 >$! Define system-wide symbols
<16:33:23.25 >$!
<16:33:23.26 >$ DIR :==DIRECTORY/DATE/SIZE=ALL
<16:33:23.27 >$ LO*GOUT :==LOGOUT/FULL
<16:33:23.28 >$ PRINT :==PRINT/NOTIFY
<16:33:23.29 >$ HOME :==SET DEFAULT SYS$LOGIN
<16:33:23.30 >$ SD :==SHOW DEFAULT
<16:33:23.31 >$ DEL*ETE :==DELETE/CONFIRM
......
<16:33:23.32 >$ SET NOPREFIX
```

If you cannot use the SET PREFIX command, you can simulate checkpointing by adding the following command to key areas of your login procedures:

```
$ WRITE SYS$OUTPUT "''F$TIME()"
```

To reduce login time focus on the following:

- Execute only the procedures that are absolutely needed. Use the F$MODE and F$GETJPI lexical functions to identify the type of user logging in and execute only the login procedures he or she needs
- Remove duplication between user login procedure and system procedures. Make sure users don't unnecessarily add commands to their own login procedures that are already executed in system or group login procedures
- Remove extensive comments from execution stream. Comments take time to execute! If you have a lot of comments (hundreds of lines) place them at the end of the command procedure but after an EXIT statement
- Use system or group-level logical names when appropriate. If all users need the same logical name, define a system or group-level logical name during system startup procedures
- Evaluate the need to migrate re-write login procedures in compiled language (see following section)

11.6.1 Compiled Login Procedures

If you define hundreds of symbols or logical names in login procedures, you may find that these commands contribute substantially to login time. In these cases, you may want to re-code symbol and logical name definitions using a compiled language (e.g., Pascal, Fortran, or Macro).

The following is a sample VAX Macro program that defines a number of global symbols using the run-time library routine LIB$SET_SYMBOL. This same strategy can be used for logical names using the routine LIB$SET_LOGICALS (for more information see Digital's *OpenVMS Runtime Library Manual*).

```
; DEFINE_SYMB.MAR
;
; Procedure to define global symbols
;
     .TITLE DEFINE_SYMB
$SSDEF                                  ; System status values
;+
; Declare external routines
;-
     .EXTRN LIB$SET_SYMBOL

     .SBTTL RO_DATA - Read Only DATA
     .PSECT RO_DATA RD,NOWRT,NOEXE

SYMBOLS:
     .ASCID "SET DEFAULT"               ; Symbol Value
     .ASCID "SD"                        ; Symbol Name
     .ASCID "$AUTHORIZE"
     .ASCID "UAF"
     .ASCID "$SYSMAN"
     .ASCID "SYSMAN"
     .ASCID "REPLY/STATUS"
     .ASCID "STATUS"

     ; add more definitions here
     ; ....
     .LONG 0                            ; End of list

GLOBAL_SYM:.LONG LIB$K_CLI_GLOBAL_SYM   ; Select Global symbols tbl

     .SBTTL CODE - Start of program
     .PSECT CODE RD,NOWRT,EXE

     .ENTRY DEFINE_SYMB ^M<>
```

```
     MOVAQ SYMBOLS,R3                Start w/ first symbol

10$:
     TSTL (R3)                      ; Check for end of table
     BEQL 20$                       ; If 0, done

     PUSHAL GLOBAL_SYM              ; Make it a GLOBAL symbol

     PUSHL R3                       ; Push address of symbol name
                                    ; Compute address of symbol value
     ADDW (R3),R3                   ; + length of symbol name
     ADDL #8,R3                     ; + 8 more bytes to skip descriptor
     PUSHL R3                       ; Push address of symbol value

     CALLS #3,G^LIB$SET_SYMBOL      ; Define it

     ADDW (R3),R3                   ; Move to next symbol name
     ADDL #8,R3                     ; + 8 bytes to skip descriptor

     BRB 10$                        ; do it again

20$: RET

     .END DEFINE_SYMB
```

Create the foregoing program using any Digital editor, then compile, link, and execute it as follows:

```
$ MACRO DEFINE_SYMB
$ LINK DEFINE_SYMB

$ SHOW SYMBOL/GLOBAL SYSMAN
%DCL-W-UNDSYM, undefined symbol - check validity and spelling

$ RUN DEFINE_SYMB
$ SHOW SYMBOL/GLOBAL SYSMAN
SYSMAN == "$SYSMAN"
```

As you can see, after DEFINE_SYMB is executed, the symbol SYSMAN is defined. To use this technique, remove equivalent DCL commands and add the RUN command to SYLOGIN.COM. For example:

```
$! SYLOGIN.COM
$!
$! ....
$ RUN SYS$MANAGER:DEFINE_SYMB
$ ...
```

Using a compiled language provides several advantages over using DCL. First, compiled code generally executes faster than comparable DCL statements as long as you need to define more than a few symbols. Second, with a compiled language, you compile symbol definitions into an executable image (e.g., DEFINE_SYMB.EXE) and install it as shareable using the OpenVMS Install utility. Once the definitions are installed, multiple users who execute the image simultaneously share the same copy in memory, reducing execution time.

```
$ INSTALL:==$INSTALL/COMMAND_MODE
$ INSTALL ADD SYS$MANAGER:DEFINE_SYMB.EXE/SHARE
```

The main disadvantages of using compiled code are that you must re-compile source code after every change and that debugging can be much more difficult than DCL. The source code for DEFINE_SYMB.MAR is located on the diskette included with the book (see Appendix E).

11.7 Establishing Menus

If you have users who are unfamiliar with DCL, you may want to configure their user login procedure as a menu command procedure. The following command procedures demonstrate this technique:

```
$! SYS$MANAGER:MENU.COM
$!
$! A simple menu system to select one of four applications
$!
$! NOTE: The symbols "123" & "WP" are defined in system login
procedures
$ wso:=write sys$output
$!
$DISP_MENU:
$ wso "1 - Word Processing"
$ wso "2 - LOTUS 1-2-3"
$ wso "3 - Mail"
$ wso "4 - Logout"
$ wso ""
$inquire/nopun inp "Enter Selection (1-4): "
$!
$ if inp .lt. 1 .or. inp .gt. 4
$ then
$ inquire/nopun inp "Invalid command, press <RETURN> to continue."
$ goto DISP_MENU
```

```
$ endif
$!
$ if inp .EQ. 1
$ then
$   DEFINE/USER SYS$INPUT SYS$COMMAND
$   WP
$   GOTO DISP_MENU
$ endif
$!
$ if inp .EQ. 2
$ then
$   DEFINE/USER SYS$INPUT SYS$COMMAND
$   123
$   GOTO DISP_MENU
$ endif
$!
$ if inp .EQ. 3
$ then
$   DEFINE/USER SYS$INPUT SYS$COMMAND
$   MAIL
$   GOTO DISP_MENU
$ endif
$!
$ if inp .EQ. 4 then LOGOUT
```

To activate the menu for a particular user, modify the UAF record as follows:

```
$ UAF
UAF> modify username /LGICMD=SYS$MANAGER:MENU.COM
```

11.8 Captive Login Procedures

In some cases, you may want to completely prevent a user from accessing DCL. This can be accomplished by developing a captive login command procedure and specifying it as the default login command procedure for the account, as follows:

```
$ UAF
UAF> modify username /LGICMD=device:[directory]CAPTIVE.COM -
      /FLAG=(CAPTIVE,DISCTLY)
```

The procedures for creating captive command procedures are discussed in chapter 17.

11.9 Additional References

Guide to Setting up a VMS System, Digital, April 1988.
Guide to System Security, Digital, June 1989.

[OpenVMS 6.0]

OpenVMS System Manager's Manual: Essentials, Digital, March 1994.

12 AUTOMATING SYSTEM MANAGEMENT

12.1 Chapter Objectives

- Use DCL batch jobs to schedule routine system management tasks
- Develop error handling and restart capabilities for automated procedures

12.2 Introduction

A number of system management tasks must be performed on a regular basis. Rather than leaving these tasks to chance, you can create batch jobs that are automatically submitted at the required time. For example, system log files (see chapter 10) should be closed and archived on a regular basis. Instead of performing this task manually, create a batch job and schedule it to run automatically once a month.

An automated batch procedure must be more robust than a DCL command procedure that you run interactively. When executing a procedure interactively, you constantly watch for error messages and decide how to proceed. Because batch jobs don't automatically notify you when there's a problem, you need to include additional code to handle error reporting and recovery.

This chapter also discusses the process of incorporating error handling and job restart capabilities into DCL command procedures.

12.3 Error Handling

Every time you execute a DCL command, OpenVMS returns a *severity level* and stores it in the symbol $SEVERITY. The severity level indicates the completion status of the executed command (Table 12.1).

Table 12.1 DCL Error Severity Levels

Severity Level	*Completion Status*
0	WARNING
1	SUCCESS
2	ERROR
3	INFORMATIONAL
4	SEVERE_ERROR (Fatal)

You can confirm this behavior by issuing the following command:

```
$ SHOW TIME                    ! Enter valid command
  15-MAY-1995 17:06:22

$ SHOW SYMBOL $SEVERITY
  $SEVERITY == "1"             ! Severity = "SUCCESS"

$ XYZ                          ! Enter invalid command

%DCL-W-IVVERB, unrecognized command verb - check validity and spelling
\XYZ\

$ SHOW SYMBOL $SEVERITY
  $SEVERITY == "0"             ! Severity = "WARNING"

$ COPY T.TMP DUA9999:          ! Copy to invalid device

%COPY-E-OPENOUT, error opening DUA9999:[SYSMGR].; as output
-RMS-F-DEV, error in device name or inappropriate device type for operation

$ SHOW SYMBOL $SEVERITY
  $SEVERITY == "2"             ! Severity = "ERROR"
```

12.3.1 Default Error Handler

By default, when a command within a command procedure generates an error with an ERROR or FATAL severity level, a message is displayed, and the command procedure immediately aborts. On the other hand, if the severity level is WARNING or INFORMATIONAL, the command procedure continues and displays an informational message. If a command generates a severity of SUCCESS, the command procedure continues without displaying a message.

This behavior is inappropriate for many automated jobs because you don't want a job to abort every time an error is encountered. If possible, you want to trap these errors and take an appropriate recovery action. In addition, some errors classified as WARNING may actually be very serious. By default, a command procedure continues executing no matter how many warnings are generated. To avoid these problems, your automated job should include a customized error handler.

12.3.2 Customized Error Handling

Establish a customized error handler by adding the following DCL command near the beginning of your command procedure:

```
$ ON severity-level THEN command
```

Then if a command within the procedure generates an error with a severity level that is greater than or equal to severity level, the specified command is executed. If you don't explicitly issue an ON ERROR statement, the default error handler is in effect and is equivalent to the following command:

```
$ ON ERROR THEN EXIT
```

To perform more elaborate error handling, you can establish a sub-routine, as follows:

```
$ ERROR_STATUS = 1              ! Set to normal completion, by default
$ ON WARNING THEN GOSUB ERROR_CHECK
$!
$! [command procedure]
$!
$! End of procedure
$ EXIT ERROR_STATUS             ! Exit and return last error status
$!
$ERROR_CHECK:
$ ERROR_STATUS = $STATUS        ! Save error status
$ ON WARNING THEN GOSUB ERROR_CHECK
$ RETURN
```

The foregoing ON WARNING statement causes the procedure to branch to an error handler (ERROR_CHECK) whenever an error occurs. The error handler saves the latest error status so that it can be later passed as a parameter to the EXIT statement, indicating the final completion status of the command procedure. The ON WARNING statement is included in the subroutine be-

cause you must reestablish a customized error handler every time it is triggered. Otherwise the default error handler is in effect. By default, when a command procedure exits, it returns the status of the last command executed. By designing your command procedures using the foregoing strategy, your command procedures will return the last error status.

Instead of trapping errors, you can disable error handling altogether by adding the following line to the beginning of your command procedures:

```
$ SET NOON                    ! Set NO ON_ERROR
```

Disabling the error handler prevents command procedures from aborting due to errors; however, it's up to you to add appropriate recovery procedures to deal with the errors. If you completely disable the error handler, you should carefully review all batch job log files either manually or using the OpenVMS Search Utility (see CHECK_STARTUP.COM in chapter 5).

12.4 Restart Handling

Batch jobs can be submitted so that they automatically restart after a system crash or other events that cause jobs to terminate abnormally. If you want batch jobs to be restartable, you must submit them with the following command:

```
$ SUBMIT/RESTART command-procedure
```

The job will then be automatically restarted if it is interrupted by any of the following events:

- A system crash *and* queue manager fail-over are not possible
- STOP/QUEUE/MANAGER/CLUSTER is executed
- STOP/QUEUE/RESET is executed
- SET ENTRY/RESTART is executed

When a restart occurs, the command procedure is started over from the beginning, and the symbols listed in Table 12.2 are defined.

To make a batch job restartable, add commands to the beginning of your command procedure that test the value of the $RESTART symbol. If $RESTART is TRUE, jump to the label stored in the BATCH$RESTART symbol. For BATCH$RESTART to be set properly, you must include SET RESTART_VALUE commands at the beginning of each major step.

Table 12.2 Batch Job Restart Symbols

Symbol	*Value*
$RESTART	Set to TRUE, which indicates a restart has occurred
BATCH$RESTART	Set to the value of the job step that was in progress when the job aborted. The job step is established from the last SET RESTART_VALUE=step command executed before the abort

For example, the command procedure BACKUP.COM performs a backup of DISK1, DISK2, and DISK3 and is submitted as follows:

```
$ SUBMIT/NOPRINT/RESTART BACKUP.COM
```

If the backup is interrupted because of a system crash, it automatically restarts whichever backup was in progress when the crash occurred.

```
$! BACKUP.COM
$!
$ IF $RESTART
$ THEN
$    IF F$TYPE(BATCH$RESTART) .NES. ""
$    THEN
$         REQUEST/TO=CENTRAL -
          "RESTARTING JOB AT STEP ''BATCH$RESTART''"
$         GOTO 'BATCH$RESTART
$    ENDIF
$ ELSE
$    REQUEST/TO=CENTRAL "NO RESTART VALUE - STARTING FROM BEGINNING"
$ ENDIF
.
.
.
$DISK1:
$ SET RESTART_VALUE=DISK1
$ BACKUP/REWIND/IMAGE DISK$USR1: MUA0:USR1.BCK
$!
$DISK2:
$ SET RESTART_VALUE=DISK2
$ BACKUP/REWIND/IMAGE DISK$USR2: MUA0:USR2.BCK
$!
$DISK3:
$ SET RESTART_VALUE=DISK3
$ BACKUP/REWIND/IMAGE DISK$USR3: MUA0:USR3.BCK
$ EXIT
```

When the above command procedure first executes, it checks to see if $RESTART is set. Because $RESTART is not set, the command procedure proceeds to the label DISK1:. Next, the procedure sets the restart value to DISK1 and begins a backup of the first disk. Assume that half way through the backup of the first disk the system crashes and automatically reboots. Because the batch job was submitted with the /RESTART qualifier, it is automatically restarted from the beginning. The symbol $RESTART is set to TRUE and BATCH$RESTART is set to DISK1 (the last restart value set). When the command procedure begins executing for the second time, it detects that $RESTART is set to TRUE, indicating that the procedure has been restarted. Next, the F$TYPE function is used to verify that the BATCH$RESTART symbol exists. If it does, the procedure issues an informational operator broadcast message and branches to the label stored in the BATCH$RESTART symbol. Because the value of this symbol is DISK1, the procedure branches to the DISK1: label and restarts the backup of DISK1.

12.5 Synchronizing Batch Jobs

In some cases, you may have automated batch procedures that depend on one another. For example, you might have a backup job that should run immediately after a disk purging job completes. Batch job dependencies like this can be handled with the DCL SYNCHRONIZE command, as follows:

```
$! NIGHTLY_BACKUP.COM
$!
$ SYNCHRONIZE NIGHTLY_PURGE    ! Wait for purge job to complete
$!
$! Backup disk 1
$!
$!...
$ EXIT

$!
$! NIGHTLY_PURGE.COM
$!
$ ERROR_STATUS = 1             ! Set to normal completion, by default
$ ON WARNING THEN GOSUB ERROR_CHECK
$!
$ PURGE ...
$ PURGE ...
$ PURGE ...
$!
$! End of procedure
```

```
$ EXIT ERROR_STATUS            ! Exit and return last error status
$!
$ERROR_CHECK:
$ ERROR_STATUS = $STATUS       ! Save error status
$ ON WARNING THEN GOSUB ERROR_CHECK
$ RETURN
```

Then, submit the two jobs as follows from another command procedure, as follows:

```
$!
$! DAILY_PROCESSING.COM
$!
$! Procedure to perform daily automated tasks-invoked by SCHEDULER
$!
$! Date: 09-Jun-93
$! ...
$! SUBMIT TASM$COM:NIGHTLY_PURGE.COM ...
$! SUBMIT TASM$COM:BCK_NIGHTLY.COM ...
$! ...
$ EXIT
```

When the two jobs are submitted, both begin executing immediately. However when NIGHTLY_BACKUP reaches the SYNCHRONIZE command it enters a wait state until the NIGHTLY_PURGE job completes.

12.6 Scheduling Batch Jobs

Once you develop robust command procedures, you need a tool that automatically submits jobs at the required frequency. SCHEDULER.COM is a command procedure that allows you to automatically submit jobs that must be run on a daily, weekly, or monthly basis. SCHEDULER submits itself every morning at 12:10 a.m. and then executes one or more of the following command procedures, if appropriate:

```
DAILY_PROCESSING     ! If the current day is Monday-Friday
WEEKLY_PROCESSING    ! If the current day is a Sunday
MONTHLY_PROCESSING   ! If the current day is the 1st day of the month
```

The source code for SCHEDULE is as follows:

```
$! SCHEDULER.COM
$!
$! Note: Define the following logicals in SYLOGICALS.COM:
```

```
$!
$! SCHED$DIR:
$!        Directory that contains the command procedures
$!        referenced in this procedure
$!
$!
$ SCHED_SUBMIT := SUBMIT/RESTART/NOPRINT/USER=SYSTEM
$ SCHED_HOUR == F$CVTIME(,,"HOUR")
$ SCHED_WKDY == F$CVTIME(,,"WEEKDAY")
$ SCHED_MTHDY == F$CVTIME(,,"DAY")
$ SCHED_MONTH == F$CVTIME(,"ABSOLUTE","MONTH") ! Use DD-MMM-YYYY format
$ SCHED_YEAR == F$CVTIME(,,"YEAR")
$!
$! See if we're being restarted
$!
$ IF $RESTART
$ THEN
$  IF F$TYPE(BATCH$RESTART) .NES. ""
$  THEN
$   REQUEST/TO=CENTRAL "RESTARTING SCHEDULER AT STEP ''BATCH$RESTART''
$   GOTO 'BATCH$RESTART
$  ENDIF
$  ELSE
$  REQUEST/TO=CENTRAL "RESTARTING SCHEDULER FROM BEGINNING."
$ ENDIF
$!
$ SET RESTART=RESUBMIT
$!
$! Re-submit for tomorrow
$!
$ SCHED_SUBMIT/AFTER="TOMORROW+00:10" SCHED$DIR:SCHEDULER
$!
$! Perform Daily Processing
$!
$DAILY:
$ SET RESTART=DAILY
$ SCHED_SUBMIT SCHED$DIR:DAILY_PROCESSING
$!
$! If Saturday, perform Weekly Processing
$!
$WEEKLY:
$ SET RESTART=WEEKLY
$ IF SCHED_WKDAY .EQS. "Saturday" THEN SCHED_SUBMIT SCHED$DIR:WEEKLY_PROCESSING
$!
$! If 1st day of Month, perform Monthly Processing
$!
$MONTHLY:
```

```
$ SET RESTART=MONTHLY
$ IF SCHED_MTHDY .EQS. "01" THEN SCHED_SUBMIT SCHED$DIR:MONTHLY_PROCESSING
$!
$ EXIT
```

The foregoing command procedure makes extensive use of the lexical function F$CVTIME to extract the components of the current date and time to determine when to execute the daily, weekly, and monthly procedures. For more information on the F$CVTIME function, refer to the *OpenVMS DCL Dictionary*.

The example MONTHLY_PROCESSING.COM included with this book moves key system log files to a directory located by the logical name TASM$ARCHIVE. For example, on August 1, 1995, this procedure moves and renames the following files:

```
SYS$MANAGER:ACCOUNTNG.DAT                          ==> TASM$ARCHIVE:ACC_AUG_1995.DAT
SYS$MANAGER:SECURITY_AUDIT.AUDIT$JOURNAL.DAT  ==> TASM$ARCHIVE:SEC_AUG_1995.DAT
SYS$ERRORLOG:ERRORLOG.SYS                          ==> TASM$ARCHIVE:ERR_AUG_1995.DAT
SYS$MANAGER:OPERATOR.LOG                           ==> TASM$ARCHIVE:OPER_AUG_1995.DAT
```

Finally, on a periodic basis (e.g., quarterly or yearly) you can archive the files to tape or delete them if they are no longer needed. The source code for the command procedures mentioned in this chapter is contained on the diskette included with this book (see Appendix E).

12.7 Tools

A number of vendors offer automated job-scheduling products that provide functionality necessary for production-oriented environments. Among the features offered in many of these products are the following:

- Menus for adding, changing, and removing jobs
- Automatic error detection
- Exception checking (e.g., late start time, late end time)
- Customizable fiscal calendars
- Programmable job dependencies

Table 12.3 is a partial list of commercially available job schedulers. See Appendix A for vendor contact information.

Table 12.3 Job Schedulers

Vendor	*Product*
Data Center Software	Q-MAN
Digital	DECscheduler
Target Systems	Target->Batch

12.8 Additional References

[OpenVMS 5.0]

Anagnostopoulos, Paul C., *VAX/VMS: Writing Real Programs in DCL*, Digital Press (1989)
VMS DCL Dictionary, Digital, June 1990.

[OpenVMS 6.0]

OpenVMS DCL Dictionary, Digital, March 1994.

13 PERFORMING BACKUPS

13.1 Chapter Objectives

- Develop and implement backup strategies that meet service-level objectives
- Minimize risks associated with on-line backups
- Automate backups with DCL command procedures
- Optimize OpenVMS backup performance

13.2 Introduction

Your system contains valuable data stored on disk. To prevent data from being lost, you should perform regular backups by copying files to some type of removable media (usually magnetic tape). If files become damaged or lost, you can recover them from backups and restore the system to a usable state.

Performing backups is very time-consuming, but like many forms of insurance, you realize its value when a problem occurs. If your home or car were destroyed and you didn't have insurance, the financial impact could be devastating. Similarly, performing anything other than robust backups might save you some time and money now, but will financially ruin you or your organization in the long run. Backups are one of your most important responsibilities—give them the attention they deserve.

Backups provide protection against the following three types of failures.

Hardware Failure

Although computer systems are relatively reliable, individual components can and do eventually fail. When failures occur, files or entire disks may be damaged or left in an inconsistent state, requiring recovery from backups.

Table 13.1 Digital RA Series Disk Annual Failure Rates

Model	*Annual Failure Rate (%)*
RA70	9.96
RA81	23.96
RA82	21.10
RA90	8.48
RA92	7.22

(Reprinted from *Reliability Ratings,* January 1992)

Usually the most severe and most frequent failures involve disk devices. Table 13.1 outlines the Annual Failure Rates for several Digital drive types. To determine the predicted failure rate for your site, multiple the quantity of each drive type by its respective failure rate. For example, if you have ten RA82 drives, expect 10 * 21%, or 2.1 failures per year. Similarly, if you have twenty RA90 drives, expect 1.7 failures per year. Of these failures, 75%–85% require either head-disk assembly (HDA) or full-unit replacement. Both require recovery from backups.

For more recent and detailed reliability statistics, contact *Reliability Ratings* (see Appendix A).

Software Failures

Data corruption can be caused by bugs in software, including OpenVMS, layered products, or in-house applications. You must perform backups so that you can restore disks to the state in which they were before corruption.

User Errors

Although usually less severe, data loss can occur if users accidentally delete files or perform incorrect updates. In these cases, you may need to perform a partial restore of one or more files from backups.

13.3 Backup Requirements

Business requirements dictate when backups can be performed; what must be backed up and how frequently; when backups should be stored off-site; and how quickly recovery must occur.

These requirements must be developed by collaborating with users and managers and considering the following issues:

- Backup window
- Backup frequency
- Recovery time
- Retention requirements
- Off-site rotation

13.3.1 Backup Window

The *backup window* is the time period during which it is acceptable for backups to be performed. For users to specify this requirement, you should make them aware of the following issues:

- When backups are performed, system performance may be severely degraded
- Some applications require that user access be restricted during backups
- You may not be able to perform backups while application-related production jobs are processing

Once you've derived an acceptable backup window, record it in terms of the exact hours of each day that backups can be performed. For example:

```
Weekdays: 00:00-06:00, 20:00-24:00
Weekends: 00:00-24:00
```

13.3.2 Backup Frequency

The data contained on a disk is an asset with an associated value. As data is manipulated, its value increases by the cost of the manipulation. For example, consider an Accounts Receivable department with 20 data entry clerks each making $15 per hour. On a normal day, clerks enter billing information and generate invoices totaling $10,000,000. If this data were lost and had to be re-entered, the cost would be as follows:

20 clerks * 8 hours * $15/hour = $2400

The value of data is also affected by the business impact of its loss. For example, if the disk containing the aforementioned invoices failed, it might

take a day to repair the disk and another day to re-enter the invoices. Because the invoices would be mailed late, the business would receive payment two days later, resulting in the following loss (assuming 10% interest):

$10,000,000 * (10%/365) * 2 days = $5479

Therefore, total value of a day's data in this example is $7879. The more quickly the value of data increases, the more frequently users require backups.

13.3.3 Recovery Time

You should discuss with users the different system failures that might occur and agree on how quickly you must be able to recover from each of them. When you calculate total recovery time, include the time to perform each of the following steps:

- Detect failure
- Determine appropriate action
- Wait for service vendor to arrive
- Diagnose problem
- Locate replacement parts
- Repair equipment
- Retrieve backups (especially if off-site)
- Restore backups
- Perform post-recovery procedures

Include every step that must be performed before users can access the system. Don't forget to include post-recovery procedures such as rerunning production jobs.

13.3.4 Archive Requirements

Backups allow you to recover the system to a previous state. However, exactly which prior states must you be able to recover to? For example, users may need to recover to specific dates in the past, such as key month-end, quarter-end, or year-end closing dates. You should coordinate your backups with these events and retain each backup long enough to meet recovery requirements.

13.3.5 Off-site Rotation

Off-site rotation provides protection against disasters that might damage disks *and* any backups stored on-site. You should determine the maximum data loss allowed in such a disaster and rotate backups off-site at an appropriate interval to meet those requirements. You should also ensure that you have the ability to retrieve backup media from off-site fast enough to meet disk recovery time requirements. Finally, you need to develop mechanisms to rotate backup media back on-site when retention periods expire.

13.4 Backup Techniques

The OpenVMS Backup utility provides the three following major backup types:

- Image backups
- Physical backups
- File backups

13.4.1 Image Backups

An *image* backup saves the entire contents of a disk to backup media, including volume initialization information. Because image backups allow you to completely restore a disk, they are considered *full backups*. The major limitation of image backups is that you *must* back up the entire contents of a disk. Similarly, when you perform an image restore, you *must* restore all files contained on the backup.

13.4.2 Physical Backups

A *physical* backup ignores the file structure of a disk and makes an exact copy of the disk on a block by block basis, including unused blocks. Normally, a physical backup of a full disk takes less time than an image backup. However, if the disk is nearly empty, a physical backup will probably take longer, because it must still copy unused disk blocks.

Physical backups have several disadvantages. You cannot selectively restore files. When you restore a physical backup, *every* block must restore perfectly; otherwise the target disk will not be usable. If you perform a physical disk backup and restore, the resulting disk structure will be exactly as it was

before the backup, including any disk fragmentation. Only an image backup and restore eliminates disk fragmentation.

Because of the serious limitations of physical backups, this chapter focuses on image and file backups.

13.4.3 File Backups

In contrast to an image backup, a *file backup* allows you to selectively copy one or more files from disk to backup media. Similarly, a *file restore* allows you to copy one or more files from backup media to disk.

You can perform a file backup of an entire disk (a full backup) by selecting all files on the disk. However, full file backups do not contain volume initialization information. You *can* completely restore a disk from a file backup, but you must manually initialize the disk before performing the restore. For this reason, it is usually more convenient to use image backups to perform full backups.

A special type of file backup is known as an *incremental file backup*. An incremental backup selectively copies all files that have changed since a previous backup. If you have disks with low file volatility, you can save considerable backup time by performing frequent incremental backups and less frequent full backups. When file volatility is high, incremental backups are not useful because an incremental backup may take as long as a full backup.

File volatility measures the total size of all files modified during a particular time period. For example, the collective size of all files that change on a daily basis might total as little as a few megabytes, or several hundred megabytes.

To calculate the daily file volatility for a particular disk, execute the following command over a period of several days:

```
$ DIR/SIZE=ALL/TOTAL/SINCE="-1-" device-name:[000000...]*.*;*
```

The resulting total indicates the total size and number of files that changed on the disk in the previous 24 hours. Depending on file volatility, you may want to perform one of the two following types of incremental backups.

Cumulative Incrementals

With *cumulative incremental* backups, you back up all files that have been modified since the last full backup. For example, you might perform a full backup on Saturday and a cumulative incremental backup Monday through

Friday evening. The Monday night backup would contain all files changed between the full backup on Saturday and Monday night. The Tuesday night backup would contain all files changed between the full backup on Saturday and Tuesday night, and so on.

To restore a disk using cumulative incremental backup, first restore the latest full backup then the most recent cumulative incremental. For example, if you used the backup strategy just described, you would restore the most recent weekend full backup and then the cumulative incremental backup from the previous night. When file volatility is very low, a cumulative incremental backup takes much less time than a full backup. However, recovery time is longer because you must restore both the full backup and the incremental backup.

Differential Backup

With *differential* backups, you backup all files changed since the last full backup *or* the last differential backup. For example, you might perform a full backup on Saturday and a differential Monday through Friday evening. The Monday night backup would contain all files changed between the full backup on Saturday and Monday night. The Tuesday night backup would contain all files changed since the Monday night backup, and so on.

To restore a disk using this backup technique, first restore the most recent full backup, then restore *every* differential backup created since the full backup. For example, if a disk failed on Thursday morning, you would first restore the full backup from the previous Saturday, then restore the Wednesday, Tuesday, and Monday differential backups (in that order). When file volatility is moderate, differential backups require less backup time. However, recovery can take much longer because you may need to perform many incremental restore operations.

13.4.4 Backup Mode

When you perform a backup, its mode is either *on-line* or *off-line.* A backup is considered on-line if the source disk is publicly mounted and user activity can possibly occur. A backup is considered off-line when the source disk is mounted privately to the backup process, preventing any user activity.

A *stand-alone* backup involves performing a system shutdown and booting the Stand-alone Backup Utility. When backups are performed from stand-alone backup, users cannot access any disks, ensuring an off-line backup.

13.4.5 Minimizing the Risks of On-line Backups

Backup integrity can be absolutely assured only when you perform off-line backups. When you perform on-line backups, you run the risk of a user's writing to a file at the exact moment it is being backed up. When this occurs, the backup of the file may be left in an inconsistent and unrecoverable state. If you need to restore from the backup, that particular file may be rendered unusable.

Unfortunately for many sites, off-line backups are simply not practical because of system up-time requirements. As long as you use the correct backup commands, you can back up files while users are reading from them. You only have to be concerned with files that are open for writing access. If recovering these files is not essential, on-line backups may be acceptable. However, if one file of a larger database does not restore properly, the entire database may be left in an inconsistent state.

You should be especially careful when performing on-line backups of an OpenVMS system disk. Most activity to a system disk is read access to executable program files, which does not affect the integrity of the backup. However, other files are written to on a regular basis, such as the error log file, accounting file, operator log file, user authorization file, audit journal file, and queue database. You can reduce the risk of inconsistent backups of these files by minimizing the following activities during on-line backups:

- System logins
- Adding/modifying user accounts
- Process termination of any kind (interactive, batch, network, subprocess, detached)
- Any batch or print job activity or submissions
- Activities that generate security alarms

[OpenVMS 5.5]

Beginning with OpenVMS 5.5, you should use the following procedure to make backup copies of the queue database files before performing an on-line system disk backup:

```
$ CONVERT/SHARE SYS$SYSTEM:QMAN$MASTER.DAT *.SAV
$ CONVERT/SHARE SYS$SYSTEM:SYS$QUEUE_MANAGER.QMAN$QUEUES *.SAV
```

13.4.6 Stand-alone Backup

Because OpenVMS requires constant access to the system disk, you cannot dismount the system disk and subsequently perform an off-line backup. To perform an off-line backup of the system disk you must use the Stand-alone Backup Utility. Invoke stand-alone backup by first performing a system shutdown and then booting the system from a Stand-alone Backup Kit you previously created on disk or tape. When in this mode, the system is running a minimal version of OpenVMS that allows you to perform only image backup and restore functions from the console terminal. Users cannot access the system, nor can you execute standard DCL commands.

If your system disk fails, you will not have access to OpenVMS or the OpenVMS backup utility. You must use stand-alone backup to perform any restores. Because stand-alone backup can process only image backups, make sure you have full image backups of your system disk. You *cannot* restore a system disk from full file backups.

Stand-alone backup commands have the same syntax as the OpenVMS backup commands, but you can only perform image backup and restore operations. The procedures for creating Stand-alone Backup Kits and booting from them are discussed in chapter 4.

13.5 Developing a Backup Strategy

To develop a backup strategy, consider the backup requirements for your organization and combine backup techniques that meet those requirements. For example, if file volatility is very high and the business requires a maximum data loss of 1 day, you can perform daily full backups. On the other hand, if file volatility is low, you can perform weekly full backups and daily cumulative incremental backups.

Table 13.2 lists a number of common backup strategies and the circumstances under which they might be appropriate.

13.6 Backup Tools

The primary tool used to perform backups is the OpenVMS Backup utility which is invoked using the following DCL command:

```
$ BACKUP input output
```

Table 13.2 Sample Backup Strategies

Backup Strategy	*File Volatility*	*Backup Window /Capacity*	*Backup Time*	*Maximum Data Loss*	*Recovery Time*
Daily full only	Very high	Unlimited /Very high	Very high	1 day	Very low
Weekly full, daily cumulative incremental	Low	Limited /Moderate	Moderate	1 day	Moderate
Weekly full, daily differential	Moderate	Very limited /Low	Low	1 day	High
Monthly full, weekly cumulative, daily cumulative	Very low	Very limited /Low	Moderate	1 day	Very high
Weekly full only	Very low	Very limited /Moderate	Low	1 week	Very high
Monthly full, weekly cumulative incremental	Very low	Very limited /Low	Low	1 week	Moderate
Monthly full, weekly differential	Very low	Very limited /Very low	Moderate	1 week	Very high
Monthly full only	Very low	Very limited /Very low	Low	1 month	Very low

When you perform a backup, the *input* is the disk being backed up and the *output* is the name of a tape drive (or other device) followed by a *save-set* name. For example:

```
$ BACKUP/IMAGE DUA1: MUA0:IMAGE.SAV
```

A save-set is a special file used by BACKUP to serve as a container for all files selected by the backup. When you perform a restore, you simply reverse the order of the input and output parameters as follows:

```
$ BACKUP/IMAGE MUA0:IMAGE.SAV DUA1:
```

The full syntax of the BACKUP command is as follows:

```
$ BACKUP/command-qualifiers -
      input /input-qualifiers output /output-qualifiers
```

The following sections detail the importance of each of the optional backup qualifiers.

13.6.1 Backup Command Qualifiers

/IGNORE=INTERLOCK

If you use this qualifier, backup attempts to copy files even if they are open. Usually you must select this option when you perform on-line backups. If you don't use this qualifier and a file is open, backup skips the file.

/IGNORE=LABEL

When you select this qualifier, backup does not perform tape label processing. This instructs backup to use a tape regardless of the current label name. If you do not select this option, backup expects the tape label to be equal to the first six characters of the input or output save-set or the label specified with the /LABEL qualifier. If there is a label mismatch, backup generates an operator message requesting further instructions.

/IMAGE

Use the /IMAGE qualifier to perform a full image backup or restore as discussed in section 13.4.1.

/INCREMENTAL

When you perform an incremental restore operation you *must* include this qualifier to indicate that the input save-set is an incremental backup.

/LABEL=label-name

As discussed earlier, this qualifier instructs backup to expect the loaded tape to contain the specified label name.

/LIST=listing-file

This qualifier causes backup to generate a file containing a complete list of all files written during the backup. By using this qualifier, you can create a data-

base of listing files that itemize the contents of each of your backups. You can then use the DCL SEARCH command to scan all or some of the listing files to find which backup contains the file for which you are looking.

/RECORD

Forces backup dates to be written to all files processed by the backup. You must use the /RECORD qualifier during full backups if you later wish to perform incremental backups (see /SINCE=BACKUP).

If you use the /RECORD qualifier only when you perform full backups, each incremental backup is a *cumulative* incremental backup. If you use the /RECORD qualifier when you perform full backups *and* with each incremental backup, each incremental backup is a *differential* backup.

/RELEASE_TAPE

This qualifier notifies backup to dismount and deallocate a tape device as soon as it is finished writing it to the tape. Use this qualifier to cause backup to automatically dismount the last tape of a backup operation (by default, you must manually dismount it). If you specify this qualifier with the /RECORD qualifier, backup frees the tape drive as soon as the backup date writing pass begins, rather than waiting for the entire backup to complete.

/VERIFY

This qualifier causes backup to perform a verification pass after a backup or restore operation. During a backup verification pass, the output save-set is reread and compared with the backup source disk. During a restore verification, the input save-set is reread and compared with the restored disk.

Although backup verification doubles the amount of time it takes to complete a backup, it is the most reliable method of ensuring backup integrity. Digital strongly recommends using /VERIFY on all backups. You may want to avoid using /VERIFY when performing on-line backups because you'll receive many verification errors regarding files that change during the backup.

13.6.2 Backup Input Qualifiers

/SINCE=BACKUP

Use this qualifier to perform incremental file backups as discussed in Section 13.4.2 (see also /RECORD).

13.6.3 Backup Output Qualifiers

/REWIND

The /REWIND qualifier is very misleading. This qualifier causes the backup media to be rewound *and* initialized. When you reuse a tape that contains a previous backup, you normally use the /REWIND qualifier. Otherwise, backup skips to the end of the backup media and appends to it. You should think of /REWIND as INITIALIZE TAPE, and /NOREWIND as APPEND TO TAPE.

To perform a backup to a medium that has never been initialized, you must use the /REWIND qualifier. If a backup requires more than one medium, subsequent media are automatically initialized. To store multiple backups on a single backup medium (such as a 4- or 8-mm tape) use the /REWIND qualifier for the first backup and the /NOREWIND qualifier for subsequent backups.

/TAPE_EXPIRATION=expiration-date

By specifying a tape expiration date you can prevent backups from being improperly overwritten. You should establish a tape expiration date on the basis of the desired retention period for the backup. For example, when you perform a backup that should be retained for 1 year, specify a tape expiration date 1 year in the future. If you attempt to use the tape before its expiration date, you receive an error message. The /TAPE_EXPIRATION qualifier is valid only when used with the /REWIND qualifier.

13.6.4 Automatic Tape Handling

When you use a tape device you normally perform the following steps:

- Allocate the device to prevent other processes from accessing the device
- Initialize the tape with a label, if necessary
- Mount the tape
- Dismount the tape when you are finished

When performing a backup, you can explicitly issue ALLOCATE, INITIALIZE, MOUNT, and DISMOUNT commands, or you can allow backup to perform most of these functions for you. When you create a backup that requires more than one tape, the backup utility can mount and dismount each tape. However, the last tape is *not* automatically dismounted, unless you use the

Table 13.3 Tape Handling Commands

Command	*Function*
ALLOCATE device-name	Allocates a device to your process, preventing other processes from using the device
INITIALIZE device-name label	Prepares a disk or tape for initial use by OpenVMS. To initialize a device, a disk or tape media must be loaded and on-line but *not* mounted. The initialization process clears a disk or tape of existing data. New media must be initialized before they can be used by OpenVMS. When invoked, INITIALIZE writes an alphanumeric label to the medium. Labels for tape devices can be 6 characters in length. Disk labels may be 12 characters long.
MOUNT device-name label log-name	Makes a volume available on a device. When a volume is mounted, OpenVMS first compares the supplied label with the actual label of the medium (as written during initialization). If the labels match, the medium is mounted and a logical name is defined that equates with the device name of the medium. By default, when you mount a tape device, it is mounted as a Files-11 formatted device. That is, you can use DIR and COPY commands to manipulate the drive.
/FOREIGN	Mounts a volume in foreign mode for use by OpenVMS backup.
DISMOUNT device-name	Performs the reverse of a MOUNT. By default, when volumes are dismounted they are also taken off-line and unloaded. Before a volume can be remounted it must be reloaded and put on-line by manual operator intervention.
/NOUNLOAD	Prevents the volume from being unloaded after dismount. The volume can then be subsequently remounted without operator intervention.

/RELEASE_TAPE qualifier. Table 13.3 lists the syntax and function of these commands. The examples in this chapter allow backup to perform these functions automatically.

13.7 Implementation

Once you develop a backup strategy, the next step is to determine the exact DCL commands necessary to implement your strategy. The following sections detail methods for performing the different backup types discussed throughout this chapter.

13.7.1 Full Backups

Full backups can be implemented by performing either a full file backup or full image backup as follows

Full File Backup

```
[load unused tape and place on-line]
$ BACKUP/REWIND/LABEL=S01W2/IGNORE=INTERLOCK -
      /VERIFY/LIST=TASM$BACKUPS:S01W2_FILEU1.LIS/RECORD -
      DISK$USER1:[000000...]*.*;* MUA0:FILEU1.BCK
```

Full Image Backup

```
[load unused tape and place on-line]
$ BACKUP/IMAGE/REWIND/LABEL=S01W1/IGNORE=INTERLOCK -
      /VERIFY/LIST=TASM$BACKUPS:S01W1_IMAGEU1.LIS/RECORD DISK$USER1: -
      MUA0:IMAGEU1.BCK
```

13.7.2 Incremental Backups

Cumulative Incremental Backup

Before you perform a cumulative incremental backup, you must first perform a full backup. During the full backup (either an image or full file backup), use the optional /RECORD qualifier. This forces the date and time of the backup to be written into the backup date field of each file after the backup completes. For example:

[Monday Full Image Backup]

```
$ BACKUP/IMAGE/REWIND/LABEL=S01W1/IGNORE=INTERLOCK -
      /VERIFY/LIST=TASM$BACKUPS:S01W1_IMAGEU1.LIS/RECORD DISK$USER1: -
      MUA0:IMAGEU1.BCK
```

Then perform the cumulative incremental backups.

[Tuesday Cumulative Incremental]

```
$ BACKUP/REWIND/LABEL=S01D2/IGNORE=INTERLOCK -
      /VERIFY/LIST=TASM$BACKUPS:S01D2_INCRU1.LIS -
      DISK$USER1:[000000...]*.*;*/SINCE=BACKUP MUA0:INCRU1.BCK
```

[Wednesday Cumulative Incremental]

```
$ BACKUP/REWIND/LABEL=S01D3/IGNORE=INTERLOCK -
      /VERIFY/LIST=TASM$BACKUPS:S01D3_INCRU1.LIS -
      DISK$USER1:[000000...]*.*;*/SINCE=BACKUP MUA0:INCRU1.BCK
```

The /RECORD qualifier is not used on successive incremental backups.

Differential Backup

The syntax for a differential backup is similar to that for a cumulative incremental backup, except each incremental backup uses the /RECORD qualifier to force an update of the file backup date.

[Monday Full Image Backup]

```
$ BACKUP/IMAGE/REWIND/LABEL=S01W1/IGNORE=INTERLOCK -
      /VERIFY/LIST=TASM$BACKUPS:S01W1_IMAGEU1.LIS/RECORD DISK$USER1: -
      MUA0:IMAGEU1.BCK
```

[Tuesday Differential Backup]

```
$ BACKUP/REWIND/LABEL=S01D2/IGNORE=INTERLOCK -
      /VERIFY/LIST=TASM$BACKUPS:S01D2_INCRU1.LIS -
      DISK$USER1:[000000...]*.*;*/SINCE=BACKUP/RECORD MUA0:INCRU1.BCK
```

[Wednesday Differential Backup]

```
$ BACKUP/REWIND/LABEL=S01D3/IGNORE=INTERLOCK -
      /VERIFY/LIST=TASM$BACKUPS:S01D3_INCRU1.LIS -
      DISK$USR1:[000000...]*.*;*/SINCE=BACKUP/RECORD MUA0:INCRU1.BCK
```

Note that the /RECORD qualifier *is* used on successive incremental backups.

13.7.3 Stand-alone Backup

Before you can boot stand-alone backup you must first perform a system shutdown (see chapter 6), as follows:

```
$ @SYS$SYSTEM:SHUTDOWN
```

During the shutdown process, do *not* request an automatic reboot. When the shutdown completes, force the console terminal into console mode by pressing the appropriate key sequence (type CTRL/P or <BREAK> or press the console HALT button). After you receive the console boot prompt, enter the appropriate console command to boot the desired Stand-alone Backup Kit. For example, to boot a kit that you previously built on the system disk, you might enter the following command:

```
>>> B/E0000000
```

The exact command necessary to boot stand-alone backup is processor-specific. Refer to Appendix C to determine the commands for your system. After you enter the correct boot command, stand-alone backup loads and displays available backup devices. You then receive the stand-alone backup prompt ($), and you can begin entering backup commands, as follows:

```
$ BACKUP/REWIND DUA0: MUA0:IMAGEA0.BCK   ! /IMAGE is implied
```

13.7.4 Full Image Restore, with Incremental Backup

The following is a sample restore of DISK$USER1 using the cumulative incrementals created in the previous examples.

[Full Image Restore of DISK$USER1 (DUA2:)]

```
[load S01W1 into tape drive]
[verify that DISK$USER1 is not mounted]
$ MOUNT/FOREIGN DUA2
$ MOUNT/FOREIGN MUA0:
$ BACKUP/IMAGE/VERIFY/LOG MUA0:IMAGEU1.BCK DUA2:
```

[Restore Cumulative Incremental]

```
[load S01D3 into tape drive]
$ BACKUP/INCREMENTAL/VERIFY/LOG MUA0:INCRU1.BCK DUA2:[*]
```

[Remount drive for public use]

```
$ MOUNT/SYSTEM DUA2: USER1 DISK$USER1
```

To restore a backup that consists of multiple incremental backups, first restore the full image backup, then restore each incremental backup starting with the latest and ending with the earliest.

13.7.5 Backup and Volume Shadowing

Digital's OpenVMS Volume Shadowing product allows you to bind two or more physical disks together and logically view them as one device called a *shadow set*. When you write to a shadow set, each operation is applied to every member of the set. In this way, the contents of all shadow members remain identical. If one members fails, it is removed from the shadow set, and users can continue accessing the remaining members.

Volume shadowing provides extra protection against drive failures; however, it is *not* a backup technique. Volume shadowing provides only limited protection against disk failure. It does not protect against failure of all members of a shadow set, system hardware failures, software failures, or user error. Even if all your disks are shadowed, you should still perform regular image or file backups.

Although volume shadowing itself is not a backup strategy, it does provide additional backup options. As on any other disk, you can perform an *on-line backup* of a shadow set, but you must consider the usual warnings regarding on-line backups, as discussed earlier in this chapter. Another option is to remove one member from a shadow set and perform an off-line backup of that member. Using this technique, you can ensure consistent backups and maintain high data availability.

The following procedure demonstrates how to back up a shadow set member:

```
$ MOUNT/SYSTEM DSA8:/SHADOW=($1$DUA8:, $1$DUA9:) USR1   ! Mount during startup
%MOUNT-I-MOUNTED, USR1 mounted on _DSA8:
%MOUNT-I-SHDWMEMSUCC, _$1$DUA8: (USR1) is now a valid shadow set member
%MOUNT-I-SHDWMEMSUCC, _$1$DUA9: (USR1) is now a valid shadow set member

$ DISMOUNT/SYSTEM DSA8:                                 ! Dissolve shadow set

$ MOUNT/SYSTEM DSA8:/SHADOW=($1$DUA8:) USR1             ! Re-create, less DUA9

$ MOUNT $1$DUA9: USR1                                   ! Mount to local process
%MOUNT-W-VOLSHDWMEM, mounting a shadow set member volume write locked

$ BACKUP/IMAGE $1$DUA9: MUA0:D2USR1                     ! Off-line backup of USR1

$ MOUNT/SYSTEM DSA8:/SHADOW=($1$DUA8:, $1$DUA9:) USR1   ! Add DUA9 to shadow set
%MOUNT-I-SHDWMEMSUCC, _$1$DUA9: (USR1) added to shadow set with a copy operation
```

To ensure a consistent backup, you must dissolve the entire shadow set then recreate the shadow set less one member. You cannot use this technique

with the backup /RECORD qualifier (e.g., for incremental backups), because when you add the member back to the shadow set, the backup date information is overwritten by the shadow copy operation.

You must also take special steps when restoring a shadow set. Backup requires that the target disk of a restore operation be mounted foreign. Digital does not support mounting a shadow set foreign, thus you must manually mount the target disk using the following commands:

```
$ DISMOUNT/SYSTEM DSA8:                                    ! Dissolve shadow set
$ MOUNT/FOREIGN/OVERRIDE=SHADOW_MEMBERSHIP $1$DUA8:        ! Mount 1 member foreign
```

then restore the disk, as follows:

```
$ BACKUP/IMAGE MUA0:D2USR1 $1$DUA8:                        ! Do restore

$ MOUNT/SYSTEM DSA8:/SHADOW=($1$DUA8:, $1$DUA9:) USR1      ! Re-create shadow set
%MOUNT-I-MOUNTED, USR1 mounted on _DSA8:
%MOUNT-I-SHDWMEMSUCC, _$1$DUA8: (USR1) is now a valid shadow set member
%MOUNT-I-SHDWMEMSUCC, _$1$DUA9: (USR1) added to shadow set with a copy operation
```

When you mount the shadow set, 1DUA8 is considered more current (because of the restore operation); thus the shadow software initiates a shadow copy from 1DUA8 to 1DUA9.

13.7.6 Media Labeling

Even if you retain tapes for a moderate amount of time you can easily accumulate a backup tape inventory containing hundreds of tapes. For even the smallest site, you'll need to adopt a tape labeling scheme that allows you to quickly locate and identify each tape. Each tape should have a written label and a unique initialized label. To keep things simple, you can make the written label the same as the initialized label. Tape labels should indicate when the backup was created and its contents. Tape labels can be only six characters long.

The sample backup procedures presented at the end of this chapter uses labels by concatenating a backup set name, the backup type, and a sequential number, as follows:

```
"Backup-set-name + Backup-type + Sequence-number"
   3 characters + 1 character + 2 characters = 6 characters
```

For example:

```
Label      Description
------     ---------------------------
S01W4      Weekly Backup #4 of Set S01
S02D2      Daily Backup #2 of Set S02
S01Y2      Yearly Backup #2 of Set S01
```

The backup set describes the set of disks contained on the tape. Some system managers include the device name in the tape label. This strategy won't work if you want to back up multiple devices onto a single tape. For example, you might organize disks into the following backup sets:

```
Backup Set: "S01"

     DISK$USER1 - DUA2
     DISK$USER2 - DUA3

Backup Set: "S02"

     DISK$SCR1  - DUA0
```

To specify the backup type, this strategy uses a single letter, as follows:

```
D - Daily
W - Weekly
M - Monthly
Q - Quarterly
Y - Yearly
```

The last two characters of the tape label are sequential numbers to make the tape labels unique.

13.7.7 Locating Backup Media

You can facilitate locating backup media by creating OpenVMS backup listing files. When you perform backups you should use the /LIST qualifier to create a list of all files written during the backup.

In the foregoing examples, all backup listing files were created in the directory TASM$BACKUPS and included the tape label and the backup save-set, as follows:

```
S01W4_IMAGEU1.LIS    ! Listing file for DISK$USER1
S01W4_IMAGEU2.LIS    ! Listing file for DISK$USER2
S02D2_IMAGES1.LIS    ! Listing file for DISK$SCR1
```

Table 13.3 Recommended UAF Values for Optimal BACKUP Performance

UAF parameter	*Recommended Value*
WSQUOTA	16384
WSEXTENT	>= WSQUOTA
PGFLQUO	32768
FILLM	128
DIOLM	4096
ASTLM	4096
BIOLM	128
BYTLM	65536
ENQLM	256

At some future time, you could search all DISK$USER2 image backups for a file named XXY.DAT, as follows:

```
$ SEARCH TASM$BACKUPS:*IMAGEU1.LIS "XXY.DAT"
```

13.8 OpenVMS Backup Performance

With the introduction of OpenVMS Version 5.2, enhancements were made to OpenVMS backup performance. However, to take advantage of these changes, you must properly configure user authorization file (UAF) parameters as indicated in Table 13.3. If you use the SYSTEM account to perform backups, changes in UAF should be made to that account.

CAUTION: If your disks are very fragmented, or if you want to minimize the impact of performing backups while users access the system, use lower values for WSQUOTA and FILLM. Increase these parameters only if you have sufficient available memory. Otherwise you may actually cause excessive paging—and therefore decrease overall backup performance.

To set these suggested UAF parameters you must ensure that the system parameters are set as follows:

```
CHANNELCNT      128             ! Must be >= FILLM
WSMAX           16384           ! Must be >= WSQUOTA
VIRTUALPAGCNT   32768           ! Must be >= PGFLQUOTA
```

Refer to chapter 18 for more detailed information on system performance issues and procedures for changing system parameters.

Table 13.4 Example Backup Command Procedures

Command Procedure	*Description*
BCK_NIGHTLY.COM	Performs backups according to a defined schedule
BCK_CALC_LABEL.COM	Determines tape label to use for a given backup using the label format discussed in this chapter
BCK_SELECT_FILE.COM	Selects a file to restore by scanning backup listing files
BCK_SCHED.COM	Generates a backup schedule that indicates the tape labels needed for each day of the month

13.9 Tools

Table 13.4 lists several example backup command procedures that are contained on the diskette included with the book (see Appendix E).

13.10 Additional References

[OpenVMS 5.0]

Guide to Maintaining a VMS System, Digital, November 1991.
VMS Backup Utility Manual, Digital, April 1988.
VMS DCL Dictionary, Digital, June 1990.
VMS Volume Shadowing Manual, Digital, June 1990.

[OpenVMS 6.0]

OpenVMS System Management Utilities Reference Manual, Digital, March 1994.
OpenVMS System Manager's Manual: Essentials, Digital, March 1994.
Volume Shadowing for OpenVMS, Digital, March 1994.

14 GETTING HELP

14.1 Chapter Objectives

- Present strategies for organizing site documentation
- Discuss Digital's support offerings
- Identify other support resources

14.2 Introduction

Sometimes when you're working on a difficult problem, you reach a point where your progress halts and you become frustrated. This usually happens because you don't completely understand some aspect of the problem. More specifically, there is something you don't know or something you once knew but have forgotten. This something prevents you from being able to develop a solution. At this point, you have two options—continue to work on the problem and become even more frustrated or try to find help.

The ability to recognize when you don't know something is important. Even more important is knowing where to find the information you need. This chapter helps you identify the resources at your disposal and helps you better use your own knowledge by organizing information into site documentation.

14.3 Developing a Site Guide

A generic data center consists of a system running a freshly installed operating system. A generic data center can be managed by just about any system manager—a generic system manager. Such a data center loses its generic status, however, the moment network equipment, customized start-up files, applications, user accounts, system management tools, batch queues, print-

ers, and terminals are added. This nongeneric site can no longer be managed by just anyone. It requires a system manager who knows all the aspects of the site that differentiate it from a generic site. Ideally, site documentation should describe all nongeneric site attributes. This information allows you and your successors to better understand the environment and allows you to resolve problems quickly. Unfortunately, collecting and organizing this information is difficult and time-consuming.

To get you started, a sample site guide outline is provided in Appendix B. As you scan each item in the outline, think about the configuration of your particular site. The information that comes to mind is what should be in the site guide. For each item ask yourself questions such as the following:

Why is it there?

What does it do?

Who uses it?

How do I manage it?

How do I monitor it?

At first you may want simply to assemble information into a three-ring binder. However, as you gather more and more information you'll probably want to make your site guide available on-line. To facilitate this process, a DCL command procedure named SYSDOC is contained on the diskette included with this book (see Appendix E). SYSDOC allows you to store documentation as standard OpenVMS text files and access them from a simple DCL menu system.

Although there is a large amount of information to document, you don't have to create an entire site guide all at once. Even the simplest documentation can be extremely helpful. For example, consider compiling a phone list of all support resources (external and internal) and the system components they support. For internal resources, include office, beeper, home, and manager phone numbers. This document might take a few hours to assemble, but it will save you an enormous amount of time. More important, such a document is essential during emergencies, when saving a few minutes is critical.

Think of the information that will save you the most amount of time. Then allocate fifteen or thirty minutes each day to work on those items. Gradually, you'll find your site guide evolving into a document that you depend on daily.

14.4 Digital Documentation

The next few sections discuss the various forms of documentation provided by Digital, including the following:

- OpenVMS Help utility (VMSHELP)
- OpenVMS Message Help utility (MSGHLP)
- OpenVMS Help text files
- OpenVMS documentation (hard copy)
- CD-ROM documentation

14.4.1 OpenVMS Help Utility (VMSHELP)

VMSHELP is a utility that provides access to OpenVMS on-line documentation. This is usually the quickest way to get answers to simple command syntax questions. Invoke OpenVMS Help using the following command:

```
$ HELP
HELP
```

The HELP command invokes the HELP Facility to display information about a command or topic. In response to the "Topic?" prompt, you can:

- Type the name of the command or topic for which you need help.
- Type INSTRUCTIONS for more detailed instructions on how to use HELP.
- Type HINTS if you are not sure of the name of the command or topic for which you need help.
- Type /MESSAGE for help with the HELP/MESSAGE utility.
- Type a question mark (?) to redisplay the most recently requested text.
- Press the RETURN key one or more times to exit from HELP.
- You can abbreviate any topic name, although ambiguous abbreviations result in all matches being displayed.

Additional information available:

```
:=              =               @               ACCOUNTING      ALLOCATE
ANALYZE         APPEND          ASSIGN          ATTACH          AUTHORIZE
AUTOGEN         BACKUP          CALL            CANCEL          CLOSE
CONNECT         CONTINUE        CONVERT         COPY            CREATE
```

```
DEALLOCATE      DEASSIGN        DEBUG           DECK            DECthreads
DEFINE          DELETE          DEPOSIT         DIFFERENCES     DIRECTORY
DISABLE         DISCONNECT      DISKQUOTA       DISMOUNT        DSNlink
DSR             DUMP            EDIT            ENABLE          ENDSUBROUTINE
EOD             EOJ             EXAMINE         EXCHANGE        EXIT
FDL             FONT            Format_of_error GOSUB           GOTO
HELP            Hints           IF              INITIALIZE      INQUIRE
INSTALL         Instructions    JOB             Kermit          LATCP
Lexicals        LIBRARY         LICENSE         Line_editing    LINK
LMCP            LOGIN           LOGOUT          MACRO           MAIL
MERGE           MESSAGE         MONITOR         MOUNT           NCP
NCS             ON              OPEN            PASSWORD        PATCH
PHONE           PRINT           PSWRAP          PURGE           Queues
READ            RECALL          RECOVER         RENAME          REPLY
REQUEST         RETURN          RMS             RTL_Routines    RUN
RUNOFF          SEARCH          SET             SHOW            SORT
SPAWN           Specify         START           STOP            SUBMIT
SUBROUTINE      Symbol_Assign   SYNCHRONIZE     SYSGEN          SYSMAN
System_Services TFF             TYPE            UIL             UNLOCK
V60_NEWFEATURES VIEW            WAIT            WRITE

Topic?
```

At the prompt, you can request more information on any of the topics presented (e.g., PRINT, SHOW).

14.4.2 OpenVMS HINTS

If you are not sure which utility or command you need information on, use the hints facility as follows:

```
$ HELP HINTS

HINTS
```

Type the name of one of the categories listed below to obtain a list of related commands and topics. To obtain detailed information on a topic, press the RETURN key until you reach the "Topic?" prompt and then type the name of the topic. Topics that appear in all upper case are DCL commands.

Additional information available:

```
Batch_and_print_jobs        Command_procedures      Contacting_people
Creating_processes          Developing_programs     Executing_programs
Files_and_directories       Logical_names           Operators_in_expressions
Physical_devices            System_management       Terminal_environment
```

```
User_environment

HINTS Subtopic?
```

At the prompt, enter the topics about which you want more information. You will be presented with a list of commands and utilities related to the topic you selected. Finally, return to the main help menu and request help on the utilities suggested.

14.4.3 Other HELP Libraries

By default, the HELP command accesses the main OpenVMS help library, which consists of the file SYS$HELP:HELPLIB.HLB. There are a number of separate help libraries for more complex utilities (e.g., Authorize) which you can list using the following command:

```
$ DIR SYS$HELP:*.HLB

Directory SYS$COMMON:[SYSHLP]

ACLEDT.HLB;1                AMDS$HELP.HLB;1          ANALAUDIT$HELP.HLB;1
ANLRMSHLP.HLB;1             CLUE.HLB;1               DBG$HELP.HLB;1
DBG$UIHELP.HLB;1            DECW$HELPHELP.HLB;1      DISKQUOTA.HLB;1
DSN$ITSHELP.HLB;2           DTEHELP.HLB;1            DTSDTR.HLB;1
EDFHLP.HLB;1                EDTHELP.HLB;1            ESS$LADCP.HLB;1
ESS$LASTCPHELP.HLB;1        EVE$HELP.HLB;1           EVE$KEYHELP.HLB;1
EXCHNGHLP.HLB;1             HELPLIB.HLB;4            INSTALHLP.HLB;1
KERMIT.HLB;                 LATCP$HELP.HLB;1         LMCP$HLB.HLB;1
MACRO$DWCI.HLB;1            MAILHELP.HLB;1           MNRHELP.HLB;1
NCPHELP.HLB;1               PATCHHELP.HLB;1          PHONEHELP.HLB;1
SDA.HLB;1                   SHWCLHELP.HLB;1          SYSGEN.HLB;1
SYSMANHELP.HLB;1            TECO.HLB;1               TFF$TFUHELP.HLB;1
TPUHELP.HLB;1               UAFHELP.HLB;1            WP.HLB;1

Total of 39 files.
```

To access one of the foregoing libraries, specify the library name using the /LIBRARY qualifier, as follows:

```
$ HELP/LIBRARY=SYS$HELP:CLUE.HLB

  Additional information available:

  CLUE

Topic?
```

14.4.4 Help Message Utility (MSGHLP)

[OpenVMS 6.0]

Beginning with OpenVMS 6.0, you can use the Help Message utility to receive context-sensitive help after receiving an error message. For example:

```
$ XYZ           ! Type and invalid command
%DCL-W-IVVERB, unrecognized command verb - check validity and spelling
\XYZ\
$ HELP/MESSAGE
IVVERB, unrecognized command verb - check validity and spelling

Facility: CLI, Command Language Interpreter (DCL)

Explanation: The first word on the command is not a valid DCL command
or a symbol name equated with a command. The rejected portion of the
command is displayed between backslashes.

User Action: Check the spelling of the command name or symbol name
and reenter the command.
```

You can also use MSGHLP at any time to display information about a specific system message, as follows:

```
$ HELP/MESSAGE NOSUCHFILE
```

For further information on MSGHLP, refer to Digitals' *OpenVMS System Messages: Companion Guide for Help Message Users.*

14.4.5 OpenVMS Help Text Files

[OpenVMS 6.0]

Beginning with OpenVMS 6.0, Digital provides the option to copy several help-related text files onto your system disk during a OpenVMS upgrade or installation. By default, these files are created in the directory specified by the logical name SYS$VMSDOC_TEXT. You can list these files and search them with the OpenVMS Search utility, as follows:

```
$ SET DEFA SYS$VMSDOC_TEXT

$ DIR/SIZE

Directory SYS$COMMON:[SYSHLP.VMSDOC]
```

```
VMSDOC_GLOSSARY.TXT;1                622
VMSDOC_MASTER_INDEX.TXT;1           5843
VMSDOC_OVERVIEW.TXT;1                180

Total of 3 files, 6645 blocks.
```

For example, if you wanted to know what an Accounting File is, you could issue the following command:

```
$ SEARCH/WINDOW=10 VMSDOC_GLOSSARY.TXT "ACCOUNTING FILE"

accounting file

A file that contains information about system resource use. The
information is stored in binary format: you cannot print it with the
PRINT command or type it with the TYPE command. To process the
accounting file, use the Accounting utility. See current accounting
file.
...
```

Then, if you want to find out which manual discusses the Accounting utility you could issue the following command:

```
$ SEARCH/WINDOW=10 VMSDOC_MASTER_INDEX.TXT "ACCOUNTING"
```

The /WINDOW=10 qualifier tells the SEARCH utility to list the 10 lines of text surrounding any matches.

14.4.6 OpenVMS Manuals

Like that of many operating systems, the OpenVMS manual set is huge. There are two basic sets of OpenVMS manuals: the Base Documentation Set and the Extended Documentation Set.

The *OpenVMS 5.0 Base Documentation Set* is meant for system managers of small stand-alone systems or for users of large systems. The Extended Set is intended for larger installations or for sites performing software development. With the introduction of OpenVMS 6.0, Digital made important documentation changes. Now, the Base Set includes a much more thorough discussion of system management tasks and utilities. This change may allow more sites to get by with only the Base Set. Table 14.1 compares the differences between the OpenVMS 5.0 and 6.0 documentation sets.

Table 14.1 Comparison Between OpenVMS 5.0 and 6.0 Documentation Sets

Documentation Set	*Contents*
OpenVMS 5.0 Base Set	*VMS Users Manual* *VMS System Manager's Manual* *VMS License Management Utility Manual* *Building Dependable Systems: The VMS Approach*
OpenVMS 6.0 Base Set	*OpenVMS User's Manual* *OpenVMS DCL Dictionary (Volumes 1 and 2)* *OpenVMS System Manager's Manual (Volumes 1 and 2)* *OpenVMS System Management Utilities Reference Manual* *OpenVMS VAX Guide to System Security (OpenVMS VAX Only)* *OpenVMS Alpha Guide to System Security (OpenVMS Alpha Only)*
OpenVMS 5.0 Extended Set	*Base Documentation Set* *User Subkit (includes Master Index)* *System Management Subkit* *Programmer Subkit*
OpenVMS 6.0 Extended Set	*Base Documentation Set* *Master Reference Kit* *Advanced System Management Kit* *Networking Kit* *Programming Kit*

To locate information contained in the Base Set, use the indexes at the end of each manual. When using the Extended Set, use the indexes included with each subkit, or use the *VMS Master Index*, which covers all subkits.

Another good starting point is the system messages manuals: *VMS System Messages and Recovery Procedures Reference Manual: Part I* and *VMS System Messages and Recovery Procedures Reference Manual: Part II*. These two manuals list almost every possible OpenVMS error message and describe the messages in detail. The descriptions in these manuals tend to be vague, but sometimes they can point you in the right direction. Beginning with OpenVMS 6.0, these manuals are not included in the Base or Extended Sets and must be

purchased separately. However, much of the same information can be accessed on-line using the MSGHLP facility, discussed earlier in this chapter.

You should also make sure that you have all the documentation necessary to support the third-party products installed at your site. Frequently, these manuals address product-specific system management tasks. Without updated manuals, you will not be able to adequately support these products.

14.4.7 CD-ROM Documentation

Digital distributes documentation on compact disk–read-only memory (CD-ROM) as a product called *Consolidated On-Line Documentation* (OLD). These disks contain the entire Extended Documentation Set and the documentation for more than 60 Digital layered products. This amounts to more than 50,000 pages of documentation. Currently, you need a DECwindows terminal (or emulator) and the BOOKREADER application to view CD-ROM documentation. However, Digital has discussed providing access using character-based terminals in the near future. Another benefit of OLD is that you can view all documentation on the disks, even for products you haven't licensed.

You can purchase CD-ROM distribution service as part of your software support agreement and receive a new OLD disk every few months. If you have many people who need access to OpenVMS documentation, CD-ROM documentation can save you a lot of time and money.

14.5 Digital Support

Digital offers four basic software support products:

- Software Support Service (SSS)
- Software Node Service (SNS)
- Layered Product Support (LPS)
- Media and Documentation Distribution Service (MDDS)

14.5.1 Software Support Service

When you purchase SSS for a specific Digital system it provides support for *system integrated products* associated with that system. System integrated products typically include the following:

- OpenVMS
- DECnet
- VMScluster Software

SSS provides the following support services:

Telephone Support

Telephone support allows three named individuals within your organization to make unlimited telephone support calls to Digital Customer Support Centers twenty-four hours a day, seven days a week.

Advanced Electronic Support

DEC Support Technical Notes

When many customers log support calls concerning a similar problem, Digital technicians publish technical notes providing a detailed solution or workaround. Upon request, Digital will mail or FAX these notes to you.

For example, the title of one available note is *Establishing a Common System Dump File in a VMScluster.*

Software Patches

Sometimes software bugs are discovered in the operating system and layered products. If the impact is severe enough, Digital issues a patch kit. These patch kits are deployed only to customers who log a support call that exhibits the symptoms repaired with the patch. Sometimes you may experience the impact of a software bug, but because you do not describe the symptoms exactly, a patch is never suggested. If you suspect such a problem, it helps to ask if patches exist for components of your problem. As you hear what patches are available, it may become obvious that you need them.

DSNlink

DSNlink provides a mechanism to log support calls via modem, directly search Digital's problem database, send mail messages to support technicians, and receive responses to support calls as OpenVMS Mail messages on your system. You can also download patch kits and software product descriptions (SPDs).

To setup DSNlink, request a start-up kit from the Customer Support Center. DSNlink is a big time-saver—it is a *must* for any system manager.

Right-to-use New Versions

Right-to-use New Versions provides you with an ongoing license granting you the right to use new versions of OpenVMS and system integrated products. When you first purchased OpenVMS, you were granted a license to use *only* the then current version of OpenVMS. These and other software licensing issues are discussed in more detail in chapter 7.

14.5.2 Software Node Service

SNS is a stripped-down version of SSS. Again, it applies only to support of OpenVMS and system integrated products. SNS provides only the right to use new versions (as described above). In general, you purchase SSS for one system at your site (to receive telephone support) and SNS for the remaining systems (to keep your OpenVMS licenses updated). You do not need to purchase telephone support for every system.

14.5.3 Layered Product Support

LPS is a service purchased on a per-product basis to extend software support to layered products. For example, if you wanted software support for the following products, you would have to purchase LPS for each one:

- VAX C C Compiler
- VAX FMS Forms Management System
- DECps DEC Performance Solution

The services to which LPS entitles you depend on the level of software support you have on the system itself. If you purchased SSS for your system, LPS provides the following services:

- Telephone support
- Advanced electronic support
- Right-to-use new versions

On the other hand, if you have only SNS, LPS provides only the right to use new versions. You cannot purchase LPS unless you have either SSS or SNS.

14.5.4 Media and Documentation Distribution Service

SSS, SNS, and LPS provide you with the right to use new versions of OpenVMS and layered products, but they do *not* provide you with the actual

software to perform the upgrades. You must purchase media and documentation for each product. To automatically receive the media and documentation kits for new product versions, subscribe to Digital's MDDS.

If you have a large number of layered products, consider using Digital's CD-ROM distribution service. For considerably less cost, you receive on compact disks media and documentation for nearly every Digital product.

14.5.5 Self-Maintenance

Self-maintenance means you have no software support agreement with Digital. This raises important licensing issues. If you do not subscribe to software support, you do not receive the right to use new versions of Digital products. If you wish to upgrade a product, you must purchase an update license to use a new version and obtain a media and documentation kit. These issues are discussed in more detail in chapter 7.

14.5.6 Digital Support Centers

Three major Digital Customer Support Centers (CSCs) provide hardware and software technical support. They are Colorado Springs, Colorado; Alpharetta, Georgia (Atlanta area); and Shrewsbury, Massachusetts. Colorado Springs provides support for high-end systems such as VAX 6000 or larger. Atlanta supports office automation products and low-end systems such as VAX 4000, MicroVAX, and VAXstations. Shrewsbury provides support for several specialized products including SNA products and Print Servers (LPS20/40).

Digital hardware and software support is referred to as Digital Services and is reached by calling a single number: (800) 354-9000. After you place your call, your request is routed to the proper support center.

To place a support call, perform the following steps:

1. Before you even think about placing a support call, you must be prepared to clearly describe the nature of the problem. If you receive a specific error message, write down the *exact* syntax of that message. If you are experiencing a general problem, record the exact set of steps that produced the problem. If at all possible, you should be able to consistently duplicate the problem.
2. Determine if you have a hardware or a software problem. If you're not sure, guess. If you have a hardware problem, first locate the serial number of the system. If you have a software problem, you'll need your software support access number. This is the number given to you by your

local Digital Contract Administrator when you established a software support contract.

3. Place a call to digital services (800) 354-9000. You will be greeted by a call handler who will ask you for your access number or serial number. Call handlers are trained to log your call and determine to which support team your call should be dispatched. Call handlers are not technicians. Be prepared to provide the following information:
   - Is the problem with OpenVMS or a layered product?
   - Name and phone number
   - OpenVMS version and product version
   - Problem statement
   - Hardware type (if applicable)

   It is critical that you convey your problem in brief, yet precise terms, otherwise you will not be directed to the correct support team.

4. Once the call is logged and assigned to a team, the call handler checks to see if anyone on the team is available. About 10% of the time someone is available and you are connected immediately. Most of the time, however, the team is busy and your problem is queued to the team. You are asked to leave your phone number so that a team member may call you back. Before you hang up, make sure you verify your phone number.

5. Usually within 30–60 minutes a technician returns your call. When you begin speaking to the technician, give a complete description of the problem. Statements such as "My machine won't boot," "It doesn't work," or "How soon can you fix it?" are not helpful. You need to tell the technician what error messages you're receiving; what may have recently occurred on the system; and how frequently the problem has been occurring. For example, "I begin to boot the machine. During the start-up procedures the message '%SYSTEM-F-NOGBL - insufficient global pages' is displayed. The start-up continues but hangs with a blank screen several minutes later."

Some technicians are immediately familiar with the problem and begin helping you troubleshoot the problem. If you get this type of reaction, you are in good hands. Other, less experienced technicians need to begin searching Digital's trouble database for specific key words or error messages that you've mentioned. These technicians can usually help you solve your problem; it just takes a bit longer.

If you believe a technician doesn't seem to understand your problem, suggest that they get help. If that doesn't work, ask to speak to the *duty manager,* and request that the call be reassigned. If you place a call and don't receive a response within a reasonable amount of time, contact the duty manager and request that the call be escalated. Continue to call back and escalate the call until you get results. You should use these tactics only when you need immediate response, such as when your system is down.

You don't necessarily have to wait until you have a specific problem before you place a support call. You can call just to ask a question or to learn more about a utility or command. If you place such calls though, let the call handler know that it is a low priority.

14.6 DECUS

Digital Equipment Corporation Users Society (DECUS) is a volunteer organization of Digital users throughout the world. The purpose of DECUS is to promote the exchange of information among users of Digital products. Membership is free and open to anyone interested. DECUS is by far the most comprehensive and cost-effective way to keep abreast of Digital's technology. The following sections discuss aspects of the DECUS organization in detail. To learn more about DECUS, refer to the contact information contained in Appendix A.

14.6.1 Local Users Groups

There are more than 150 DECUS Local Users Groups (LUGs) throughout the United States. LUG activities vary widely; most groups have monthly meetings and provide a newsletter. At each meeting, speakers (other DECUS members, Digital employees, or third-party vendors) present technical information on topics that members of the LUG have requested. The LUG is meant to be your primary local resource. By attending LUG meetings you can gain valuable information and meet other Digital users in your neighborhood with whom you can share problems and solutions.

14.6.2 Special Interest Groups

A Digital Special Interest Group (SIG) is a group of DECUS members who focus on a specific industry area, field of study, or application. The following are the currently active SIGs:

Artificial Intelligence
Business Applications
Datatrieve/4GL
DAARC (Data Acquisition, Analysis, Research and Control)
Data Management Systems
Electronic Publishing
Graphic Applications
Technical, Hardware and Engineering
Languages and Tools
MUMPS
Networks
Office Automation
Personal Computer
RSTS
RSX/IAS
RT-11
Site Management
Unisig (UNIX/ULTRIX Sig)
VAX Systems
Third Party Providers

Contact names for the various SIGs are available from the DECUS membership office.

14.6.3 National Symposia

DECUS organizes two week-long symposia each year—one on the east coast in the spring and one on the west coast in the fall. During a symposium, DECUS members, many of whom are senior Digital engineers, give technical presentations on topics relating to Digital equipment. Usually more than 1000 hours of presentations are scheduled and attendance exceeds 5000 DECUS members. Most presentations are 30–60 minutes long with a question and answer period at the end. Presentations vary in depth and quality, but with so many sessions to choose from it's hard not to learn something that makes the trip worthwhile.

Considering the amount of information available and the reasonable cost, DECUS is by far the best training value for the money. Many system managers consider yearly symposium attendance mandatory. The symposia, how-

ever, should be thought of as supplements to, not replacements for, formal training. Presentations are relatively unstructured, and you cannot expect to learn what you might in a focused five-day training class.

The symposia are preceded by presymposium seminars that do provide one- and two-day, focused presentations on specific topics. The seminars are frequently abbreviated formats of standard Digital training classes, taught by the same instructors, but at considerably less cost.

DECUS sponsors local seminars and regional conferences in major metropolitan areas throughout the year. The topics discussed are similar to the content of the presymposium seminars.

14.6.4 DECUS Software Library

DECUS maintains an extensive library of public domain software developed by DECUS members and other organizations. For a detailed listing of available packages, contact the DECUS office and ask for a copy of the most current *DECUS Software Library Catalog*. Software from the DECUS Library may be purchased from DECUS or obtained, usually free of charge, from a LUG in your area.

14.6.5 DECUServe

DECUServe is an on-line electronic conferencing system which allows one-on-one discussions with DECUS users from across the U.S. Hundreds of very specific topics are discussed, such as: Performance, third-party products, and Terminal server tips.

14.6.6 Master's Lists

Several SIGs and LUGs maintain lists of *masters*. Masters are people who have volunteered to provide general help on specific topics. This program is meant as a self-help tool, and you should be prepared to ask specific technical questions if you decide to contact a master.

14.7 Conclusion

You have extensive support resources at your disposal: books, magazines, Digital, and DECUS. You pay a lot of money for some of them—use them and use them often.

The average person forgets more in a day than he or she learns. To make use of past knowledge, you must record it before it's forgotten. Documentation allows members of a system management team to share information and provides a permanent resource to their successors. Even the best system manager with the most elaborate tools is not truly effective without good site documentation.

15 SYSTEM INTEGRITY MONITORING

15.1 Chapter Objectives

- Detect system problems before they affect users
- Explore Digital's error analysis tools
- Monitor disks for low space and device errors
- Receive automatic notification when queues become stopped or stalled

15.2 Introduction

Many problems that occur within a system are not preventable. For example, disks run out of space; hardware fails; and print queues stall. When any one of these events occurs, users may be unable to access their applications. Many of these problems don't just happen spontaneously; they develop over time. By performing vigilant system integrity monitoring, you frequently can detect a developing problem before users are affected. You should intercept problems as they exhibit symptoms, not just react to their effects.

This chapter discusses system events that are clues to pending problems. A number of DCL command procedures are included with this book that provide automatic problem detection and reporting.

15.3 Monitoring Hardware Errors

Whenever hardware errors occur, the error formatter (ERRFMT) detached process writes an error record to the system error log file. The location of the error log file (ERRLOG.SYS) is specified by the logical name SYS$ERRORLOG, which by default is defined as follows:

```
SYS$SPECIFIC:[ERRORLOG]
```

The error log file contains a record of all system and device errors that occurred since the log file was created. Unfortunately, ERRFMT does not send a notification message when device errors occur. Therefore, you must monitor hardware errors using one or more of the following utilities.

15.3.1 SHOW ERROR

The SHOW ERROR command displays a count of all device errors that occurred since the last system boot, as follows:

```
$ SHOW ERROR

Device      Error Count
PTA0:          1
MUA0:          9
DUA0:          3
```

The results of SHOW ERROR can be misleading—some devices always indicate an error count of 1 or 2. Frequently these are not actual errors but normal controller initialization messages that occur during system startup. For example, the foregoing error for PTA0 is only an initialization message. You should perform a SHOW ERROR immediately after a system boot so you'll know which devices log initialization events.

If you see error counts increasing for a device, you should contact your hardware service vendor to see if repairs are necessary. Sometimes, even one error can indicate a serious problem that requires immediate attention.

15.3.2 ANALYZE/ERROR

The OpenVMS ANALYZE utility is a tool for reporting errors contained in the error log file. The syntax for the ANALYZE utility is as follows:

```
$ ANALYZE/ERROR/qualifiers
```

where *qualifiers* are one or more of the following:

/SINCE=date

The error log file may contain device errors for many months or even years. To reduce the length of a report, use the standard date selection qualifiers as follows:

```
$ ANALYZE/ERROR/SINCE=YESTERDAY
$ ANALYZE/ERROR/SINCE=01-Jan-1993
```

/SUMMARY

By default, a full page (or more) of information is presented for each error log entry. This level of detail is mainly for Digital service personnel. To get a brief report, use the following command:

```
$ ANALYZE/ERROR/NOFULL/SUMMARY
```

/INCLUDE=*component*

To report errors on a specific system component, use the following commands:

```
$ ANALYZE/ERROR/INCLUDE=MEMORY
$ ANALYZE/ERROR/INCLUDE=DUA0
```

/OUTPUT=*filename*

It is usually helpful to send the results of ANALYZE/ERROR to a file, as follows:

```
$ ANALZE/ERROR/OUTPUT=ERROR.RPT
```

15.3.3 VAXsim Plus

VAXsim Plus (VAX System Integrity Monitor) is a utility that facilitates hardware error analysis and notification. Digital distributes VAXsim free of charge to customers who have a Digital Hardware Maintenance agreement. If you are covered by such an agreement, you may already have these tools installed. If not, ask your Digital representative to install them. VAXsim components include the following.

Analyzer

The VAXsim Analyzer provides an interactive, graphical interface to your system error log file. When you invoke the analyzer, boxes that represent system devices are drawn on the screen. Devices with error counts that exceed predefined thresholds are highlighted. When you select a specific device, you are given a single line message indicating the type and quantity of errors for the

device. This is much easier then reading a full-page output from ANALYZE/ERROR. In a VMScluster environment VAXsim displays the hardware errors for the entire cluster.

Fault Manager

The VAXsim Fault Manager consists of a detached system process, VAXSIM-V.VVV (where V.VVV is the VAXsim version), which continuously monitors and analyzes errors as they occur. If error rates on a device exceed predefined thresholds, a fault notification message is dispatched. A component of the Fault Manager is a notification database, which indicates which OpenVMS users should be notified when faults occur. These messages are sent either by OpenVMS Mail or as broadcast messages via the REPLY command.

Repair Manager

The Repair Manager is used to keep track of repairs made by Digital. Whenever a device is repaired, you or your hardware service technician should make an entry into the repair database. The repair log not only helps you track when repairs were made but also affects VAXsim error analysis.

15.4 Monitoring Disks

Even when space is carefully managed, disks run out of space. This can occur if a user accidentally creates a large file or when a software problem creates a large log file. Running out of disk space has a severe effect on users. Users trying to save documents might lose their work, or other key system tasks might fail in the middle of a large process, leaving files or databases in corrupted states.

To prevent these problems, you should periodically monitor disk space using the SHOW DEVICE command as follows:

```
$ SHOW DEVICE D          ! Specify "D" to view all disk devices

Device          Device    Error     Volume          Free    Trans      Mnt
 Name           Status    Count     Label         Blocks    Count      Cnt
BIRCH$DUA0:     Mounted       0     VAXVMSV055     17070       10        1
BIRCH$DUA1:     Mounted       0     VAXVMST060     78699      147        1
```

If you see disk space drop below 10% of available space, you should take immediate action to reclaim space as discussed in chapter 10. You should also

give special attention to any disks that are system disks—if your system disk runs out of space the system can hang or crash.

An example command procedure named CHECK_DISKS, which automates disk space monitoring, is discussed at the end of this chapter.

15.5 Monitoring Processes

A number of system detached processes should always exist (e.g., JOB_CONTROL, OPCOM, ERRFMT). Sometimes OpenVMS bugs cause these jobs to fail, and you have to manually restart them. Even the failure of JOB_CONTROL is not immediately apparent—users can't login, and batch jobs can't be submitted, but other processing continues unaffected. It might take hours before you notice that a problem has occurred.

You should periodically use the SHOW SYSTEM command to verify that key system processes are running, as follows:

```
$ SHOW SYSTEM

VAX/VMS T6.0-0E7 on node BIRCH 20-MAY-1995 12:27:46.24 Uptime 0 01:27:42

Pid      Process Name    State      Pri  I/O       CPU       Page flts  Ph.Mem
00000021 SWAPPER         HIB         16    0    0 00:00:03.44         0       0
00000025 IPCACP          HIB         10    7    0 00:00:01.66       124      71
00000026 ERRFMT          HIB          8   82    0 00:00:01.14       236      86
00000027 OPCOM           HIB          8   65    0 00:00:01.45       409      35
00000028 AUDIT_SERVER    HIB         10   69    0 00:00:02.32       590      73
00000029 JOB_CONTROL     HIB         10   46    0 00:00:01.22       685      97
0000002A QUEUE_MANAGER   HIB          9   86    0 00:00:03.34       944     285
0000002B PRINT_MGR       HIBO         9    -- swapped out --              487
0000002C NETACP          HIBO         9    -- swapped out --              295
0000002D EVL             HIB          6   46    0 00:00:02.51      1135      49    N
0000002E REMACP          HIBO         8    -- swapped out --               36
0000002F LATACP          HIB         14   11    0 00:00:01.67      1471      87
00000031 SYSTEM          CUR          7 2918    0 00:00:55.43      6748     307
```

An example command procedure named CHECK_PROCESSES, which automates process monitoring, is discussed at the end of this chapter.

Sometimes bugs occur in products or applications that cause processes to loop forever. Such a problem can go undetected for hours, even days, consuming huge amounts of system resources and having a severe impact on system performance and response time. Detecting this condition is difficult

because you must be able to differentiate between excessive and normal system usage. Unfortunately, you probably can't automate this process, but you should periodically use the Monitor utility as follows:

```
$ MONITOR PROCESSES/TOPCPU          ! Check CPU activity

                    VAX/VMS Monitor Utility
                    TOP CPU TIME PROCESSES
                        on node BIRCH
                     20-MAY-1995 12:17:27

                      0         25         50         75         100
                      + - - - - + - - - - + - - - - + - - - - + - - -+
00000032 CPU_HOG     78  ********************************
                      |         |          |          |          |
00000031 SYSTEM       7  **
                      |         |          |          |          |
                      |         |          |          |          |
                      |         |          |          |          |
                      |         |          |          |          |
                      |         |          |          |          |
                      |         |          |          |          |
                      + - - - - + - - - - + - - - - + - - - - + - - -+
```

If you see a particular process that seems to be using an inordinate amount of CPU utilization (e.g., CPU_HOG), you can investigate that process in more detail, as follows:

```
$ SHOW PROCESS/CONTINUOUS/ID=32

                  Process CPU_HOG                        12:18:32

State             COM             Working set                300
Cur/base priority 2/1             Virtual pages             1841
Current PC        801C47AE        CPU time        00:00:01:07.11
Current PSL       01800000        Direct I/O                 351
Current user SP   7FF19BC4        Buffered I/O                68
PID               00000032        Page faults                421
UIC               [SYSTEM]        Event flags           60000001
                                                        80000000

BIRCH$DUA1:[SYS0.SYSCOMMON.][SYSEXE]SHOW.EXE
```

After you issue the SHOW PROCESS command, the foregoing information is displayed and updated every few seconds. Whenever the process runs an executable image, the name of the image is displayed at the bottom of the screen. In this case, CPU_HOG is doing something with the show command. You can also monitor the CPU time, direct I/O, and page faults counter to gauge demand of the process for CPU, disk I/O, and memory resources. Also, if you notice that the current PC (program counter) frequently indicates the same addresses, you might suspect that the process is in some kind of infinite loop.

You can further gauge what a process is doing by listing all the files opened by the process using the System Dump Analyze (SDA) utility, as follows:

```
$ ANALYZE/SYSTEM
VAX/VMS System analyzer

SDA> SHOW SUMMARY              ! List all active processes

Extended Indx Process name     Username        State Pri PCB       PHD       Wkset
--PID -- ---- ------------     --------        ----- --- ---       ---       -----
00000021 0001 SWAPPER                          HIB    16  802B13A0  802B1200      0
00000032 0004 CPU_HOG          SYSTEM          COM     1  803A4800  80683600    313
00000025 0005 IPCACP           SYSTEM          HIB    10  803982C0  805C8800     71
00000026 0006 ERRFMT           SYSTEM          HIB     8  8036DAC0  80763A00     86
00000027 0007 OPCOM            SYSTEM          HIB     8  80397EC0  80638A00     42
00000028 0008 AUDIT_SERVER     AUDIT$SERVER    HIB    10  803986C0  8065E000     73
00000029 0009 JOB_CONTROL      SYSTEM          HIB     8  80397AC0  8073E400    101
0000002A 000A QUEUE_MANAGER    SYSTEM          HIB     9  80397CC0  806A8C00    306
0000002B 000B PRINT_MGR        SYSTEM          HIBO    9  8036E680  00000000    487
0000002C 000C NETACP           DECNET          HIBO    9  8036F600  80613400    295
0000002D 000D EVL              DECNET          HIB     6  803960C0  806F3800     48
0000002E 000E REMACP           SYSTEM          HIBO    8  80396E00  00000000     36
0000002F 000F LATACP           SYSTEM          HIB    14  80399940  80718E00     85
00000031 0011 SYSTEM           SYSTEM          CUR     7  803CB000  805EDE00   1354
```

To select the CPU_HOG process, specify its index from the above list, as follows:

```
SDA> SET PROCESS/INDEX=4
```

You can then use the following command to view all the files CPU_HOG has open:

```
SDA> SHOW PROCESS/CHANNELS          ! Show open I/O channels

                    Process active channels
                    -----------------------

Channel Window      Status    Device/file accessed
------- ------      ------    --------------------
0010    00000000              DUA1:
0060    80378480              DUA1:[VMS$COMMON.SYSEXE]DCL.EXE;1 (sect)
0080    80395F40              DUA1:[VMS$COMMON.SYSLIB]DCLTABLES.EXE;8)
0090    8038AD00              DUA1:[SYS0.SYSMGR]T.LOG;8
00A0    803AEAC0              DUA1:[SYS0.SYSMGR]T.COM;25
```

CPU_HOG appears to be executing a DCL command procedure (T.COM) and sending output to a log file (T.LOG). You can use the DCL TYPE command to display the contents of the command procedure and log file. Beginning with OpenVMS 6.0, you can also use a modified TYPE command to continuously display the last few lines of an active log file, as follows:

```
$ TYPE/CONTINUOUS/TAIL SYS$MANAGER:T.LOG

...
...
$10:
$ SHOW TIME
  20-MAY-1995 13:05:27
$ goto 10
$10:
$ SHOW TIME
  20-MAY-1995 13:05:27
$ goto 10
$10:
$ SHOW TIME
  20-MAY-1995 13:05:27
$ goto 10
```

As you can see, the CPU_HOG process is trapped in an infinite loop, repeatedly displaying the system time.

You can use additional monitor commands to watch for processes performing inordinate disk I/O activity. For example, the following display indicates a process that is issuing 16 disk I/O operations per second:

```
$ MONITOR PROCESSES/TOPDIO          ! Check Disk I/O activity

                    VAX/VMS Monitor Utility
                TOP DIRECT I/O RATE PROCESSES
                        on node BIRCH
                     20-MAY-1995 12:25:16

                          0         5         10        15        20
                          +---------+---------+---------+---------+
00000042 DISK_HOG     16  **********************************
                          |         |         |         |         |
                          |         |         |         |         |
                          |         |         |         |         |
                          |         |         |         |         |
                          |         |         |         |         |
                          |         |         |         |         |
                          |         |         |         |         |
                          +---------+---------+---------+---------+
```

If this activity continued for more than a few minutes, you can use SHOW PROCESS and ANALYZE/SYSTEM to determine exactly what images or command procedures the activity is executing.

Unfortunately, there really are no absolute guidelines for detecting excessive system activity. You should regularly monitor your system to define normal activity. When you detect activity that exceeds your own norms, you should investigate individual processes in detail, using the techniques discussed in this section.

15.6 OPCOM

The Operator Console Messenger (OPCOM) detached process is the primary facility used to notify you of key system events. When a system event occurs, OpenVMS formats a message and requests that OPCOM deliver it. To receive OPCOM messages, you must be at an operator console. By default, only the system console (OPA0) is considered an operator console. However, you can manually enable any terminal by using the following command:

```
$ REPLY/ENABLE
```

You should include this command in your LOGIN.COM so that any terminal to which you log in is automatically enabled to receive OPCOM messages.

Table 15.1 Example System Integrity Monitoring Procedures

Command Procedure	*Function*
CHECK_QUEUES.COM	Uses the Get Queue Information lexical function (F$GETQUI) to scan active print and batch queues. When queues are found to be in a stopped, stalled, or paused state, broadcast messages are sent to the operator console.
CHECK_PROCESSES.COM	Uses the Get Job Process Information lexical function (F$GETJPI) to monitor for the existence of a set of expected processes. You control which processes are scanned by creating a text file that contains a list of process names (e.g., JOB_CONTROL, OPCOM). When a process contained in the list you specified is no longer detected, broadcast messages are sent to the operator console terminal.
CHECK_DISKS.COM	Uses the Get Device Information lexical function (F$GETDVI) to determine the available free space and error count for each disk you specify. When free space or error counts reach the thresholds you specify, broadcast messages are sent to the operator console.

Because OPCOM sends notifications using *broadcast* messages, you must also ensure that your terminal is broadcast enabled by issuing the following command:

```
$ SET TERMINAL/BROADCAST
```

15.7 Tools

Table 15.1 lists several command procedures that are located on the diskette included with this book (see Appendix E).

15.8 Additional References

[OpenVMS 5.0]

VMS Error Log Utility Manual, Digital, April 1988.

[VMS 6.0]

OpenVMS System Manager's Manual: Tuning, Monitoring, and Complex Systems, chapter 17, "Getting Information about the System," Digital, March 1995.

16 MAIL MANAGEMENT

16.1 Objectives

- Identify critical OpenVMS Mail components
- Configure mail logical names
- Compress user mail files
- Fix message count mismatch problems
- Create and maintain distribution lists

16.2 Introduction

This chapter focuses on OpenVMS Mail system management tasks and techniques for using Mail as a system management tool. In environments where mail is heavily utilized, you must perform a number of mail management tasks on a regular basis. Otherwise system performance suffers and excessive disk space is consumed.

16.3 System Profile Database

The system profile database contains an entry for each OpenVMS Mail user that describes how mail options are configured and the user's new message count. You can view a user's entire profile only by executing the SHOW ALL command from the user's account, as follows:

```
MAIL> SHOW All
Your mail file directory is SYS$SYSROOT:[SYSMGR.MAIL].
Your current mail file is SYS$SYSROOT:[SYSMGR.MAIL]MAIL.MAI;1.
No folder is currently selected.
The wastebasket folder name is WASTEBASKET.
```

```
Mail file SYS$SYSROOT:[SYSMGR.MAIL]MAIL.MAI;1
      contains 9080 deleted message bytes.
You have 8 new messages.

You have not set a forwarding address.
Your personal name is "MORON".
Your editor is TPU.
CC prompting is disabled.
Automatic copies to yourself are disabled.
Automatic deleted message purge is enabled.
Your default print queue is SYS$PRINT.
You have not specified a default print form.
```

By default, the system profile database is located on the system disk as follows:

```
SYS$SYSTEM:VMSMAIL_PROFILE.DATA
```

Depending on your configuration, the profile database should be located in either the SYS$SPECIFIC or SYS$COMMON component of SYS$SYSTEM. On a stand-alone system VMSMAIL_PROFILE should be located in the SYS$COMMON:[SYSEXE]. Within a VMS cluster you can maintain a single profile database for the entire cluster or a separate database for each cluster member (in SYS$SPECIFIC:[SYSEXE]). However, because mail profiles are directly associated with user accounts, you should use the same configuration that you established for your user authorization database (see chapter 9).

OpenVMS assumes that VMSMAIL_PROFILE is located in either of the SYS$SYSTEM roots (common or specific). However, you should define a system-wide logical name to explicitly select one of the paths. For example, to establish a single common profile database, create the following logical name:

```
$ DEFINE/SYSTEM/EXECUTIVE VMSMAIL_PROFILE -
      SYS$COMMON:[SYSEXE]VMSMAIL_PROFILE.DATA
```

If your system disk is heavily utilized, you may wish to locate the profile database to an alternative disk as follows:

```
$ DEFINE/SYSTEM/EXECUTIVE VMSMAIL_PROFILE -
      DISK$USER1:[SYSFILES]VMSMAIL_PROFILE.DATA
```

Wherever you choose to locate the profile database, add the logical name definition to the startup procedure SYLOGICALS.COM (see chapter 5).

16.4 User Mail Database

The text of mail messages is stored in each user's personal mail database, which by default is located in the user's default login directory. The user's mail database consists of a MAIL.MAI file and possibly a number of files of the format MAIL$xxxx.MAI. Short mail messages (less than 2000 characters) are stored in the file MAIL.MAI. Longer messages are stored in separate MAIL$xxxx.MAI files (where xxxx is a unique character string).

As users receive messages, their default login directory tends to fill up with mail-related files. To minimize file clutter, you may want to move each user's mail database to a subdirectory as follows:

```
[from User's account]
$ SET DEFAULT SYS$LOGIN        ! go to home directory
$ MAIL
MAIL> SET MAIL_DIRECTORY [.MAIL]
MAIL> EXIT
```

When a user deletes a long message, its associated MAIL$xxxx.MAI file is automatically deleted. However, when smaller messages (those stored in MAIL.MAI) are deleted, space in MAIL.MAI is not automatically reclaimed. To reclaim this space you must periodically compress each user's mail file, as follows:

```
MAIL> COMPRESS DSK$SYST:[SIMPSONB]MAIL.MAI

%MAIL-S-CREATED, DSK$SYST:[SIMPSONB]MAIL_0070_COMPRESS.TMP;1 created

%MAIL-S-COPIED, DSK$SYST:[SIMPSONB]MAIL.MAI;1 copied to
    DSK$SYST:[SIMPSONB]MAIL_0070_COMPRESS.TMP;1 (2 records)

%MAIL-S-RENAMED, DSK$SYST:[SIMPSONB]MAIL.MAI;1 renamed to
    DSK$SYST:[SIMPSONB]MAIL.OLD;1
%MAIL-S-RENAMED, DSK$SYST:[SIMPSONB]MAIL_0070_COMPRESS.TMP;1 renamed to
    DSK$SYST:[SIMPSONB]MAIL.MAI;1

MAIL> EXIT
```

When compression is completed, the old MAIL.MAI file is renamed to MAIL.OLD. If the compression was successful, delete the MAIL.OLD file, as follows:

```
$ DELETE DSK$SYST:[SIMPSONB]MAIL.OLD;
```

See the Tools section of this chapter for a sample command procedure (MAIL_COMPRESS) that compresses every mail file on a specific disk.

16.5 Managing Mail Profiles

16.5.1 Viewing Profiles

With the proper privileges enabled (SYSPRV), you can use the Mail utility to view or modify any user's forwarding address and personal name, as follows:

```
$ MAIL
MAIL> SHOW FORWARD/USER=username
MAIL> SET FORWARD/USER=username
MAIL> SHOW PERSONAL_NAME/USER=username   ! SET/USER is not allowed
```

Other profile attributes can be viewed or modified only from the user's account.

You can use wild cards to view multiple profiles as follows:

```
MAIL> SHOW FORWARD/USER=*/ALL

Username                Forwarding address
BALDWIN                 None
DSN$SERVER              None
SMITH                   SYSTEM
SYSTEM                  BALDWIN
TEMP1                   None
```

16.5.2 Adding Profiles

A mail profile is automatically added for each user the first time he or she invokes the Mail utility; thus it is not normally necessary to add profiles manually. However, you may want to add profiles for nonexistent users to provide automatic forwarding to existing mail accounts or to provide transparent forwarding to remote systems.

Normally, to send mail to users on a remote system you must precede the username with the remote DECnet node name. For example,

```
[from node BIRCH::]
MAIL> SEND
To: MOSCOW::YELTSIN
Subj: Yo, Boris!
^Z
```

This process can be simplified by establishing a dummy profile on node BIRCH:: for user YELTSIN. Even though YELTSIN does not have an account on node BIRCH, a mail profile can be added with a forwarding address, as follows:

```
[on node BIRCH::]
$ MAIL
MAIL> SET FORWARD/USER=YELTSIN MOSCOW::YELTSIN
MAIL> EXIT
```

Now, users on node BIRCH can send mail to YELTSIN without having to specify the remote DECnet node name.

16.5.3 Removing Profiles

When you remove a user account with the Authorize utility, you should also remove the user's mail profile, as follows:

```
$ UAF
UAF> REMOVE username
UAF> EXIT

$ MAIL
MAIL> REMOVE username
```

WARNING: Make sure you delete the mail profile entry when you delete an account from SYSUAF. Otherwise, if an account with the same name is ever added, it may inherit previous mail profile settings.

16.6 Mail Logical Names

The system logical name MAIL$SYSTEM_FLAGS is used to customize aspects of the mail system in VMScluster environments. MAIL$SYSTEM_FLAGS can be set to a value between 0 and 7, as shown in Table 16.1.

Define the logical name to equate to the sum of the options you want to activate. For example, to activate all options, define the following logical name:

```
$ DEFINE/SYSTEM/EXECUTIVE MAIL$SYSTEM_FLAGS "7"
```

To make this change effective permanently, add the command to the site-specific startup procedure SYLOGICALS.COM (see chapter 5).

Table 16.1 Setting MAIL$SYSTEM_FLAGS

Flag Value	*Effect*
1	Deliver mail directly to cluster accessible disk rather than using DECnet
2	Send users new mail broadcast messages cluster-wide
4	Include message deliver time in new mail broadcast messages

16.7 Forcing Message Count Synchronization

Sometimes a user's new mail message count indicates that there is new mail when actually there is not. This problem occurs when the system profile database (VMSMAIL_PROFILE) is out of synchronization with one or more user mail databases (MAIL.MAI). This phenomenon occurs when the system and user mail databases are on separate disks and a disk failure occurs. For example, if a user disk fails on Friday and is restored from backups that were taken on Monday, the system profile database reflects the number of new mail messages that existed on Friday, not on Monday.

To resolve this problem, have any affected users read all new mail until there are no new messages. Then have them issue the following commands:

```
$ MAIL

You have 1 New mail message.

MAIL> READ/NEW            ! forces re-synchronize
MAIL> EXIT
$
$ MAIL

Should say "You have no New messages."
```

16.8 Managing Mail Distribution Lists

A distribution list is a text file with a file extension of *.DIS* that contains a list of OpenVMS Mail addresses. You can then specify a distribution list in response to the standard Mail *To:* prompt instead of having to enter each address manually. When you create a distribution list, enter valid usernames (or other distribution lists) each on a separate line, as follows:

```
$ EDIT MAIL_LISTS:ALLTECHS.DIS          ! MAIL_LISTS is a logical name
!
! Exclamation points (!) are for comments
!
! ALLTECHS.DIS - all technical staff
!
@MAIL_LISTS:ALLOPS                  ! All operations staff
! Others Misc.
JONESB
SMITHJ
TORESSA
^Z
```

To send a message to all users on the distribution list, issue the following commands:

```
$ MAIL
MAIL> send
To: @MAIL_LISTS:ALLTECHS.DIS
Subj: Hello
[Enter message]
^Z
```

You can further simplify this process by creating a system-wide logical name in SYLOGICALS.COM as follows:

```
$ DEFINE/SYSTEM/EXEC      DIS_ALLTECHS      "@MAIL_LISTS:ALLTECHS.DIS"
```

Once the logical name is defined, enter DIS_ALLUSERS in response to the To: prompt. Start all mail distribution list logical names with a common prefix (e.g., DIS_) so that you can view all distribution lists easily, as follows:

```
$ SHOW LOGICAL DIS_*
```

16.9 Using MAIL as a Management Tool

The OpenVMS Mail utility can be used as a simplistic trouble-tracking system as follows. Establish a dummy mail profile, PROBLEM, which is forwarded to your account as follows:

```
$ MAIL
MAIL> SET FORWARD/USER=PROBLEM BALDWINL
MAIL> EXIT
```

Then notify users to send mail to PROBLEM whenever they need help. Any mail they send will be forwarded to your mail account.

When you receive problem requests from users, the requests are added to your NEWMAIL folder. Create the following folders, which you can use to set your schedule:

- REVIEWED problems you have reviewed but not scheduled
- CRITICAL problems you need to resolve as soon as possible
- TODAY problems to resolve today
- THISWEEK problems to resolve this week
- THISMONTH problems to resolve this month

After you read a new message, issue the following command to file it into the appropriate scheduling folder:

```
MAIL> READ/NEW
MAIL> FILE folder-name
```

To read messages in a particular folder, issue the following commands:

```
MAIL> SELECT folder-name
MAIL> READ
```

Create the following distribution lists for groups of people to whom you frequently send mail:

DIS_OPERATORS

DIS_PROGRAMMERS

DIS_SYSMGR

For example, mail is sent to the DIS_SYSMGR list after system startup (see chapter 5) to indicate problems with startup procedures. Similarly, the system monitoring utilities in chapter 15 send mail to DIS_SYSMGR.

16.10 Tools

The following DCL command procedure compresses all MAIL.MAI files contained on a particular disk:

```
$!
$! MAIL_COMPRESS.COM
$!
$! Created: 19-Apr-1993 L. Baldwin
$!
$! Desc:
$! Command procedure to compress all MAIL.MAI files on the disk
$! specified by the parameter P1
$!
$ disk = P1
$Loop:
$ temp = f$search("''disk':[000000...]MAIL.MAI")
$ if temp .eqs. "" then goto CLEAN_UP
$!
$ write sys$output "Compressing ''temp"
$ define FILE "''temp'"
$ mail
compress FILE
exit
$ goto LOOP
$!
$ CLEAN_UP:
$ delete/log 'disk':[000000...]MAIL.OLD;
```

Invoke MAIL_COMPRESS as follows:

```
$@MAIL_COMPRESS DISK$USER1     ! Do not include a colon (:)
```

WARNING: This procedure deletes all copies of file MAIL.OLD.

The source code for MAIL_COMPRESS is located on the diskette included with this book (see Appendix E).

16.11 Additional References

[VMS 5.0]

VMS Mail Utility Manual, Digital, April 1988.

[VMS 6.0]

OpenVMS System Manager's Manual: Essentials, chapter 6, "Managing Mail Accounts," Digital, March 1994.

OpenVMS User's Manual, chapter 6, "Mail," Digital, March 1994.

17 SECURITY MANAGEMENT

17.1 Chapter Objectives

- Ensure system access to authorized users
- Enforce password content standards
- Discuss UIC-based and ACL-based object protection
- Adjust security-related system parameters
- Consider network security requirements
- Review procedures for auditing security events

17.2 Introduction

Like performing backups, security management is a system management task that is easily neglected. Because a system that is not secure will still operate, it's easy to focus your efforts on more pressing tasks. The increase in publicized system break-ins and computer virus incidents has increased sensitivity to security matters, but these events tend to encourage a dictatorial approach to security management (e.g., "This is my system and thou shalt not enter").

Although a large part of an effective security management policy is to restrict, you have to be careful not to lose sight of the service objectives of your organization. You want to deny access to *unauthorized* users, but you also want to ensure access to *authorized* users. As you implement security policies, remember that any service denial to valid users is system downtime.

Unless specifically noted, the discussions within this chapter are relevant to OpenVMS VAX 5.5 and OpenVMS AXP 1.5. New and changed features introduced with OpenVMS VAX 6.0 are summarized in Table 17.1. As of this writing, version 6.0 features are not available on AXP systems. However, this situation is expected to change in future releases of OpenVMS AXP.

Table 17.1 OpenVMS 6.0 New and Changed Features

OpenVMS 6.0 New Feature	*Description*
Security domain distributed cluster-wide	Previously, to change the security attributes of a device or volume, you had to enter the command on all cluster members. Now, changes are automatically distributed
Security templates	Previously, the default security attributes for system objects were hard-coded within OpenVMS code. Now, user-modifiable security templates perform this function. Security templates also eliminate the need to reapply access control lists (ACLs) on some system objects after every system boot
New SET SECURITY command	Replaces various commands that configured security and attributes for system objects
ALARM_JOURNAL changes	ACE (access control entry) used to generate an OPCOM security alarm *and* an audit event to be created in the audit journal file. ALARM_JOURNAL has been renamed to ALARM and only generates an OPCOM broadcast. New ACE added (AUDIT) that generates an entry in audit journal.
Audit server data file renamed	The file that defines the configuration for the AUDIT_SERVER process has been renamed from AUDIT_SERVER.DAT to VMS$AUDIT_SERVER.DAT
New auditable events	Any object access, use of privileges, network connections, system parameter changes, use of identifiers, use of process control system services, UAF database changes, system time changes
New privileges	AUDIT, DOWNGRADE, IMPORT, UPGRADE
New system parameters	SECURITY_POLICY
Protected subsystems	Allow users to temporarily acquire identifiers while running an executable image

17.3 Controlling System Access

Before anyone can access an OpenVMS system, a username and password must be presented and validated against the user authorization file (UAF), as discussed in chapter 9. Thus, the UAF is your first line of defense against unauthorized access. To ensure that user accounts are not compromised, you should consider the following:

- Password content and length
- Access restrictions
- Captive accounts

17.3.1 Password Content and Length

OpenVMS passwords are encoded using a one-way encryption scheme and stored in the UAF. Anyone with read access to SYSUAF.DAT can view encrypted passwords. However, there is no way to directly decrypt them. There are only two ways you can learn someone else's account password:

- If the account owner reveals it
- By repeatedly trying all possible password combinations until a match is found

Therefore, to prevent user accounts from being compromised, you must ensure that passwords are difficult to guess, and you must minimize the likelihood that users will reveal passwords to others. Unfortunately, these two requirements conflict with each other, requiring a strategy that achieves a careful balance. Passwords must be difficult to guess, but not so hard that users write them down. When developing a password protection strategy, you should consider the following issues.

System Passwords

Normally, when users initiate a login, they immediately receive the standard "Username:" and "Password:" prompts. You can add an additional level of password protection by establishing a *system password*, as follows:

```
$ UAF
UAF> MODIFY/SYSTEM_PASSWORD=XYZZY
```

Next, configure each terminal device that you want to enable for system passwords by adding the following commands to system startup procedures (e.g., STARTUP_SETTERM.COM):

```
$ SET TERMINAL/PERMANENT/SYSPASSWORD TXA0:
$ SET TERMINAL/PERMANENT/SYSPASSWORD TXA1:
etc.
```

When a terminal port is enabled for the system password, an interactive login proceeds as follows:

```
<RETURN>          ! User presses <RETURN> key
<BELL>            ! System responds with an audible bell ONLY
XYZZY<RETURN>     ! User must enter system password
Username:         ! If system password is valid, normal login begins
Password:         ! If username and password valid, access is granted
```

You can enable system passwords on all terminal devices (including LAT and remote terminals) by setting bit 19 of the SYSGEN parameter TTY_DEFCHAR2, as follows:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> PARAMETER SHOW TTY_DEFCHAR2

Parameter Name              Current
--------------              -------
TTY_DEFCHAR2                4098

SYSMAN> EXIT

$ TEMP = 4098               ! Save current value
$ TEMP[19,1] = 1            ! Set Bit 19
$ SHOW SYMBOL TEMP          ! Display new value

TEMP = 528386  Hex = 00081002 ...
```

Finally, change the value of TTY_DEFCHAR2 to the calculated value using AUTOGEN, as discussed in chapter 18.

Multiple Passwords

You can configure user accounts so that they require both a primary and a secondary password, as follows:

```
$ UAF
UAF> MODIFY username/PASSWORD=(primary-pwd, secondary-pwd)
```

When a user initiates a login that requires multiple passwords, the login proceeds as follows:

```
Username: SIMPSONB          ! Enter Username
Password: xxxxxxxxxx        ! Enter primary password (not echoed)
Password: yyyyyyyyyy        ! Enter secondary password (not echoed)
```

You may want to consider establishing secondary passwords for all accounts that have system privileges.

Minimum Password Length

To make passwords difficult to guess, you should ensure that all passwords are at least six characters long. By default, the minimum password length for system accounts is eight; for all other accounts it is six. To increase the num-

Table 17.2 Number of Password Combinations

Length	*Using A–Z*	*Using A–Z, 0–9*
1	26	36
2	676	1296
3	17,576	46,656
4	456,976	1,679,616
5	11,881,376	60,466,176
6	308,915,776	2,176,782,336

ber of combinations, you also should encourage users to include numeric digits in their passwords. As you can see from Table 17.2, there are more than 2 billion possible combinations for a password six characters long (when numbers and digits are used).

You should not specify a password length lower than the default. To increase password length, use the following Authorize command:

```
UAF> MODIFY username/PWDMINIMUM=min-length
```

Password Lifetime

The password lifetime for an account determines how frequently the password must be changed. For example, if the password lifetime for an account is 90 days, the password must be changed at least once every 90 days. You should configure accounts with password lifetimes of no more than 30–90 days. The default password lifetime is 90 days. To change the password lifetime for an account (e.g., to 60 days), issue the following command:

```
UAF> MODIFY username/PWDLIFETIME="60-00:00"
```

Locked Passwords

If you must ensure that passwords adhere to strict content standards, use *locked* passwords. With locked passwords, only the system manager can set passwords with the Authorize utility. It's usually not practical to enable locked passwords for too many accounts because *you* have to change passwords every time they expire. Locked passwords are most appropriate for accounts shared by many users because you don't want one person changing the password and potentially locking out other users.

To configure accounts for locked passwords, use the following command from Authorize:

```
UAF> MODIFY username/FLAG=LOCKPWD
```

Generated Passwords

Another method of enforcing password content is to use *generated* passwords. When an account is configured with generated passwords, users must select new passwords from a list generated by OpenVMS. Generated passwords comprise a variable number of semi-random alphabetic characters and are at least as long as the minimum password length for an account. Generated passwords do not form words, but they are constructed so that they can be pronounced.

To enable generated passwords for a specific account, issue the following command:

```
UAF> MODIFY username/FLAG=GENPWD
```

If a user is enabled for generated passwords, and the user attempts to change the password, the user receives the following prompts:

```
$ SET PASSWORD

Old password:                          ! Enter old password
                (Pronunciation)
cehokbej        ce-hok-bej
eajhaumda       eaj-haum-da
vufholid        vuf-ho-lid
elfnawwra       elf-naw-wra
nacseavwoa      nacs-eav-woa

Choose a password from this list, or press RETURN to get a new list

New password:                ! Must select one from above list
```

Although they defeat password guessing, generated passwords almost guarantee that users will write down their passwords because the passwords are nearly impossible to remember. As a result, forcing generated passwords can actually increase the risk of a security breach. If your site requires strict password content, consider implementing a custom password filter in place of generated passwords, as discussed later in this section.

Password History

By default, OpenVMS maintains a history of the passwords previously used by each account and prevents them from being used again for a certain period of time. Password history data is stored in the following file:

```
SYS$SYSTEM:VMS$PASSWORD_HISTORY.DATA
```

By default, OpenVMS tracks up to the last 60 passwords used by each user (history limit) and retains each password for 365 days (history lifetime). If a user enters more passwords than can be stored in the history file, the user is forced to use generated passwords.

You can modify the values for history limit and history lifetime by adding the following logical names to system startup procedures:

```
$ DEFINE/SYSTEM/EXEC SYS$PASSWORD_HISTORY_LIMIT "50"     ! 50 passwords
$ DEFINE/SYSTEM/EXEC SYS$PASSWORD_HISTORY_LIFETIME "120"    ! 120 days
```

You can also disable password history for a specific user, using the following command:

```
UAF> MODIFY username/FLAG=DISPWDHIS
```

Password Dictionary

By default, users cannot select passwords contained in the Digital-supplied password dictionary. The password dictionary contains about 50,000 words and is stored in the following file:

```
SYS$LIBRARY:VMS$PASSWORD_DICTIONARY.DATA
```

You can disable dictionary checking for a specific user account by using the following authorize command:

```
UAF> MODIFY username/FLAG=DISPWDDIC
```

Custom Password Filter

You can use a high-level language (e.g., C or Fortran) to implement a custom password filter that forces users to select passwords that meet specific criteria. To accomplish this you must define a function POLICY_PLAINTEXT that receives a username and a password as parameters. Within your function, you can verify that passwords meet the requirements for your site (e.g., must have

at least one numeric character, cannot be an anagram of the username). You can also use the $GETUAI (get user account information) system service to read parameters from the UAF and perform additional checks.

To enable a password filter, perform the following steps:

```
$ CC VMS$PASSWORD_POLICY            ! If written in C, for example
$ @SYS$EXAMPLE:VMS$PASSWORD_POLICY_LNK
$ COPY VMS$PASSWORD_POLICY.EXE SYS$LIBRARY:
$ INSTALL:==$INSTALL/COMMAND
$ INSTALL ADD SYS$LIBRARY:VMS$PASSWORD_POLICY/OPEN/HEAD/SHARE

$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> PARAMETER USE ACTIVE
SYSMAN> PARAMETER SET LOAD_PWD_POLICY 1
SYSMAN> PARAMETER WRITE ACTIVE
SYSMAN> EXIT
```

To make the change permanent use AUTOGEN to change the LOAD_PWD_POLICY parameter.

Several example password filters are included with OpenVMS and located in the SYS$EXAMPLES directory (VMS$PASSWORD_POLICY.*). The source code for an expanded password filter (VMS$PASSWORD_POLICY.C) is located on the diskette included with this book (see Appendix E).

WARNING: If you don't need custom password filters, verify that LOAD_PWD_POLICY is disabled. Otherwise, this mechanism can be used to trap and record valid username and password combinations.

17.3.2 Access Restrictions

You can restrict certain types of system access to specific days of the week or specific times during each day. For example, you can prevent interactive logins on weekends or dial-up access during the evening. You enforce these restrictions on an account by account basis.

To use access restrictions you must first define the primary and secondary days of the week for each user account, as follows:

```
UAF> MODIFY username/PRIMEDAYS=(day, day, day ...)
```

To indicate a day as a primary day, enter MONDAY, TUESDAY, etc. To configure a day as a secondary day, enter NOMONDAY, NOTUESDAY, etc. By default, all user accounts are configured so that Monday through Friday are primary days and Saturday and Sunday are secondary days, as follows:

```
/PRIMEDAYS=(MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, NOSATURDAY, NOSUNDAY)
```

Once you define the primary days, grant or restrict access to each account using the following authorize qualifiers:

```
/ACCESS, /BATCH, /DIALUP, /INTERACTIVE, /LOCAL, /NETWORK, /REMOTE
```

For example,

```
UAF> MODIFY username/access-restriction

/ACCESS=(PRIMARY, 7-18)   ! Allow any access 7 A.M. to 6 P.M. (M-F)
/ACCESS=(7-19)            ! Allow any access 7 A.M. to 7 P.M. (M-S)
/NOBATCH=(7-18)           ! Prohibit batch access 7 A.M. to 6 P.M. (M-S)
/NODIALUP=(SECONDARY)     ! Prevent dial-up access on weekends
```

17.3.3 Captive Accounts

Captive accounts allow you to restrict users to running a particular DCL command procedure (a captive command procedure) without the ability to access the DCL. In this way, you can restrict users to the functions you provide within the captive procedure.

To make an account captive, set the CAPTIVE login flag and establish a captive login command procedure, as follows:

```
$ UAF
UAF> MODIFY user-account /FLAGS=CAPTIVE/LGICMD=device:[directory]CAPTIVE_MENU.COM
```

When a user logs into a captive account, the following rules are in effect:

- CTRL/Y interrupts or errors force an immediate logout
- The DCL INQUIRE command is disabled
- SPAWN is disabled

To prevent automatic logout, a captive login command procedure must include a customize error and interrupt handler, as follows:

```
$ ON ERROR THEN GOTO label
$ ON CONTROL_Y THEN GOTO label
```

You must also use the READ command instead of INQUIRE because the later cannot be used in a captive login procedure. The following is an example of a captive login procedure that implements a simple menu system. It is contained on the diskette included with this book (see Appendix E):

```
$! CAPTIVE_MENU.COM
$!
$! A simple CAPTIVE menu system to select one of four applications
$!
$ on error then goto DISP_MENU
$ on CONTROL_Y then goto DISP_MENU
$ set control=(t,y)                ! Enable interrupts
$ wso:==write sys$output
$!
$DISP_MENU:
$ wso "1 - Word Processing"
$ wso "2 - LOTUS 1-2-3"
$ wso "3 - Mail"
$ wso "4 - Logout"
$ wso ""
$ read/end=DISP_MENU/prompt="Enter Selection (1-4): " sys$command inp
$!
$ if inp .lt. 1 .or. inp .gt. 4
$ then
$ read/end=DISP_MENU/prompt= -
     "Invalid command, press <RETURN> to continue." SYS$COMMAND INP
$ goto DISP_MENU
$ endif
$!
$ if inp .EQ. 1
$ then
$  DEFINE/USER SYS$INPUT SYS$COMMAND
$! Invoke WP application here
$  GOTO DISP_MENU
$ endif
$!
$ if inp .EQ. 2
$ then
$  DEFINE/USER SYS$INPUT SYS$COMMAND
$! Invoke 123 application here
$  GOTO DISP_MENU
$ endif
$!
$ if inp .EQ. 3
$ then
$  DEFINE/USER SYS$INPUT SYS$COMMAND
$  MAIL
$  GOTO DISP_MENU
$ endif
$!
$ if inp .EQ. 4 then LOGOUT
```

17.4 Object Protection

After a user successfully performs a login, *object protection* mechanisms provide the next layer of access control. With object protection, a *subject* (a process acting on behalf of a user) attempts access to an *object* (a file, for example). Access is granted if the protection of the object allows it; otherwise, access is denied.

The following more commonly used object protection mechanisms are discussed in this chapter:

- Volumes (magnetic tape and disk)
- Devices (disk, magnetic tape, terminals)
- Files and directories
- Queues
- Security class—New OpenVMS 6.0

For information on the other objects (capabilities, common event flag clusters, global sections, logical name tables, and resource domains), refer to the OpenVMS manuals listed at the end of this chapter.

You protect objects using either of the following object protection types:

- User identification code (UIC)–based protection
- Access control list (ACL)–based protection

By default, access to all system objects is controlled solely by UIC-based protection. To enable ACL-based protection, you have to manually add ACLs to each object. When an object is protected by both UIC- and ACL-based mechanisms, the following precedents are in effect:

1. ACLs take priority over UIC-based protection
2. If an ACL grants access, access is granted and UIC-based protection is *not* evaluated
3. If an ACL explicitly denies access, access is denied, unless UIC-based protection grants access through the SYSTEM or OWNER access categories

Access categories are discussed in the following sections.

17.4.1 UIC-based Protection

With UIC-based protection the following three factors determine access.

Table 17.3 Valid UIC Formats

UIC Type	*Format*	*Example*
Numeric UIC	[group-number, member-number]	[100,2]
Alphanumeric UIC	[group-name, member-name]	[BILLING,SIMPSONB]
Alphanumeric UIC	[member-name]	[SIMPSONB]
Alphanumeric UIC	member-name	SIMPSONB

Subject Owner

After a user obtains access to the system by performing a login, a process is created on the user's behalf. The user's process acts as the *subject*. The subject owner is the UIC of the process attempting access. For example, after a user logs in to the account SIMPSONB, a process is created and assigned the UIC value [SIMPSONB] from the user's account record. Any object access by the user is considered to be initiated by the subject owner: [SIMPSONB].

Depending on how the user's account is configured, the subject owner's UIC may be expressed in one or more of the formats listed in Table 17.3.

To determine the valid UIC formats for a specific user, view the user's account entry using the Authorize utility, as follows:

```
UAF> SHOW SIMPSONB

Username: SIMPSONB
                          UIC: [100, 2] ([SIMPSONB])
```

All values listed to the right of "UIC:" can be used as UIC values. If the account has an alphanumeric UIC, it can be used without the square-brackets ([ ]).

Object Owner

Every object has an associated owner, as specified by a UIC value. For example, the file T.TMP (a file object) might be owned by [SIMPSONB], as indicated by the following command:

```
$ DIRECTORY/OWNER T.TMP

Directory DISK$USER1:[SIMPSONB]

T.TMP;1          [SIMPSONB]          ! Object owner UIC

Total of 1 file.
```

Beginning with OpenVMS 6.0, the SHOW SECURITY command can be used in place of the DIRECTORY command, as follows:

```
$ SHOW SECURITY T.TMP

DISK$USER1:[SIMPSONB]T.TMP object of class FILE
     Owner: [SIMPSONB]
     Protection: (System: RWED, Owner: RWED, Group: RWED, World: RE)|
     Access Control List: <empty>
```

Object Protection Mask

Each object has an associated *protection mask* that specifies how the object can be accessed. A protection mask lists four *access categories* (SYSTEM, OWNER, GROUP, and WORLD) and specifies the type of access that is allowed to each category. Protection masks are formatted as a list of access categories and access types separated by a colon character (:), as follows:

```
(SYSTEM:rwed, OWNER:rwed, GROUP:rwed, WORLD:rwed)  ! Full format
(S:rwed, O:rwed, G:rwed, W:rwed)                   ! Abbreviated format
```

For example, a file might have an associated protection mask, as follows:

```
$ DIRECTORY/PROTECTION T.TMP

Directory DISK$USER1:[SIMPSONB]

T.TMP;1    (RWED, RWED, RWE, RE)

Total of 1 file.
```

In the foregoing example, the characters RWED specify read, write, execute, and delete access types. A fifth access mode, CONTROL, is not explicitly specified in the protection mask but is implicitly granted if one or more of the following conditions are met:

- Subject holds the SYSPRV privilege
- Subject's UIC is a system UIC
- Subject's UIC exactly matches object's UIC
- Subject holds GRPPRV privilege and the subject's UIC group matches the object's

Table 17.4 UIC-based Access Categories

Example Subject UIC	*Example Object UIC*	*Relationship*	*Qualified Access Categories*
[1,4]	Any	Subject UIC is a system UIC	SYSTEM, WORLD
[100,2]	[100,2]	Same UICs	OWNER, GROUP, WORLD
[100,2]	[100,*]	Same UIC group	GROUP, WORLD
[100,2]	[300,1]	Different UICs	WORLD

When a subject accesses an object, it qualifies for one or more access categories based on the relationship between the subject's UIC and the object's UIC, as illustrated in Table 17.4.

If a subject qualifies for more than one access category, it receives the benefit of the access types from each category. The actual meaning of each access type depends on the object being manipulated. For example, WRITE access to a directory means you can add files to the directory. However, WRITE access to a queue means you can submit jobs to the queue. Some objects have different access types (e.g., logical and physical), and thus protection masks for those devices are use different characters (e.g., L and P). If a subject holds certain privileges, access capabilities may be amplified.

Beginning with OpenVMS 6.0, some access types have been renamed to better reflect their function. For example, the WRITE access type for queues has been renamed SUBMIT. As a result, you should specify the S character to allow SUBMIT access to a queue. The changes between OpenVMS 5.5 and 6.0 are detailed in Sections 17.4.3 through 17.4.7

17.4.2 ACL-based Protection

The most important limitation of UIC-based protection is that access to only four general categories can be controlled (SYSTEM, OWNER, GROUP, and WORLD). If you have more complex access control requirements, ACL-based protection provides greater flexibility.

With ACL-based protection the two following factors determine access:

- Rights identifiers held by the subject
- ACLs associated with an object

Subjects are identified by the rights identifiers they hold. Object ACLs then grant or restrict access to subjects holding given rights identifiers.

Rights Identifiers

The three following types of rights identifiers are used with ACL-based object protection:

UIC Identifiers

By default, a process holds the UIC identifiers associated with its account entry. For example, a process created by the account SIMPSONB holds the numeric UIC identifier [100,2] and alphanumeric UIC identifier [SIMPSONB].

```
UAF> SHOW SIMPSONB

Username: SIMPSONB
...                             UIC: [100, 2] ([SIMPSONB])
```

General Identifiers

The real power of ACL-based protection centers on *general identifiers*. General identifiers are arbitrary names containing 1 to 31 characters with at least one alphabetic character. They are created as follows:

```
$ UAF
UAF> ADD/ID ANNUAL_REPORT
UAF> ADD/ID BILLING_DATA
UAF> ADD/ID CORP_OFFICER
```

You make users holders of a general identifier by *granting* them the Authorize utility, as follows:

```
UAF> GRANT/ID identifier username
UAF> GRANT/ID CORP_OFFICER SIMPSONB
```

General identifiers can also have several associated attributes, as listed in Table 17.5.

To enable attributes, specify them when you create the identifier *and* when you grant it, for example:

```
UAF> ADD/ID/ATTRIBUTE=(RESOURCE,DYNAMIC) CORP_OFFICER
UAF> GRANT/ID/ATTRIBUTE=(RESOURCE,DYNAMIC) CORP_OFFICER SIMPSONB
```

Table 17.5 Identifier Attributes

Identifier Attribute	*Function*
RESOURCE	Allows a holder of the identifier to charge disk space usage to the identifier
DYNAMIC	Allows the identifier to be dynamically enabled or disabled after login with the DCL SET RIGHTS command
NO_ACCESS (new OpenVMS 6.0)	Prevents identifier from affecting access. Used to allow disk space to be charged to an identifier
HOLDER_HIDDEN (new OpenVMS 6.0)	Prevents users from displaying holders of an identifier, unless they themselves hold the identifier
NAME_HIDDEN (new OpenVMS 6.0)	Prevents users from displaying an identifier name, unless they themselves hold the identifier
SUBSYSTEM (new OpenVMS 6.0)	Allows holders to create and maintain a protected subsystem (see Section 17.6)

Environmental Identifiers

A number of *environmental identifiers* are pre-defined and automatically associated with a process depending on its login mode:

```
BATCH          ! Held by batch processes
NETWORK        ! Held by network processes
INTERACTIVE    ! Held by interactive processes
LOCAL          ! Held by processes attached to local terminal ports
DIALUP         ! Held by processes attached via Dial-up port
REMOTE         ! Held by processes created with SET HOST
```

Access Control Lists

You establish ACL-based access control to an object by associating an ACL with the object. An ACL consists of one or more access control entries (ACEs) enclosed in parentheses, as follows:

```
(ace)
(ace)
(ace)
.
.
.
```

For example, a directory might have the following associated ACL:

```
$ DIRECTORY/ACL T.TMP

Directory DISK$USER1:[SIMPSONB]

T.TMP;1
  (IDENTIFIER=CHILD,ACCESS=READ+WRITE+EXECUTE+DELETE)    ! <== ACE #1
  (DEFAULT_PROTECTION,S:RWED,O:RWED,G:RE,W)              ! <== ACE #2
  (ALARM_JOURNAL, ACCESS=WRITE+DELETE+FAILURE)           ! <== ACE #3

Total of 1 file.
```

As the foregoing example demonstrates, there are three types of ACEs, as follows.

Identifier ACE

An Identifier ACE grants or restricts access to subjects who hold specific rights identifiers, using the following format:

```
(IDENTIFIER=identifier, OPTIONS=option, ACCESS=access-modes)
```

Identifier can be any rights identifier, including the following:

- UIC identifier (e.g., [SIMPSONB])
- General identifier (e.g., CORP_OFFICER)
- Environmental identifier (e.g., BATCH)

For example, to grant full access to the file REPORT.TXT to users who hold the general identifier ANNUAL_REPORT, perform the following steps:

1. Create and grant a general identifier using authorize

```
$ UAF
UAF> ADD/ID ANNUAL_REPORT
UAF> GRANT/ID ANNUAL_REPORT SIMPSONB
UAF> GRANT/ID ANNUAL_REPORT SIMPSONH
```

2. Associate an ACE with the file granting access to the identifier

```
$ SET ACL/OBJECT=FILE/ACL= -
(IDENTIFIER=ANNUAL_REPORT,ACCESS=READ+WRITE+EXECUTE+DELETE) REPORT.TXT
```

You can also deny access to REPORT.TXT from all dial-up users, by adding the following ACE:

```
$ SET ACL/OBJECT=FILE/ACL=(IDENTIFIER=DIALUP,ACCESS=NONE) REPORT.TXT

$ DIRECTORY/ACL REPORT.TXT

Directory DISK$USER1:[SIMPSONB]

REPORT.TXT;1
      (IDENTIFIER=ANNUAL_REPORT,ACCESS=READ+WRITE+EXECUTE+DELETE) ! <== ACE #1
      (IDENTIFIER=DIALUP,ACCESS=NONE)                             ! <== ACE #2

Total of 1 file.
```

When you add ACEs that deny access, you have to make sure that you order them to produce the desired effect. OpenVMS scans ACEs from top to bottom. As soon as it encounters an ACE that grants access, it grants access and stops scanning. Similarly, as soon as it finds an ACE that denies access, it denies access and stops scanning. The ACL in the example does not deny access to all dialup users. Users who access the file from dialup and who hold the ANNUAL_REPORT identifier are granted access because OpenVMS encounters that ACE first. If you wanted to deny access to all dialup users, the DIALUP ACE would have to be placed before the ANNUAL_REPORT ACE, as follows:

```
$ DIRECTORY/ACL REPORT.TXT

Directory DISK$USER1:[SIMPSONB]

REPORT.TXT;1
      (IDENTIFIER=DIALUP,ACCESS=NONE)                             ! <== ACE #1
      (IDENTIFIER=ANNUAL_REPORT,ACCESS=READ+WRITE+EXECUTE+DELETE) ! <== ACE #2

Total of 1 file.
```

When specifying UIC identifiers within ACLs, you can grant or deny access to several users by using wild-card formats, such as the following:

[*,*]	Any UIC
[200,*]	All UICs in group 200
[PARENTS,*]	All UICs in PARENTS group

Default Protection ACE

A *default protection* ACE specifies the default protection for files created within a directory and can only be associated with directory objects.

Default protection ACEs have the following format:

```
(DEFAULT_PROTECTION,S:rwedc,O:rwedc,G:rwedc,W:rwedc)
```

The *rwedc* options correspond to UIC-based access modes (READ, WRITE, EXECUTE, DELETE, and CONTROL). The CONTROL access type can be explicitly specified.

Use default protection ACEs when you want to ensure that files within a directory are created with a specific protection mask. Normally, the protection of newly created files is taken from the default protection of the process creating the file. This can cause security problems if users change their default protection to an unacceptable value. For example, users do not require any special privileges to perform the following command:

```
$ SET PROTECTION=(W:RWED)/DEFAULT
```

Now, the user's default protection is set for world full access. Any new file this user creates could be fully accessible by all users on the system. If a directory has an associated default protection ACE, any new files created in the directory are assigned the protection mask specified in the ACE, thus overriding the default protection of the creating process.

For example, to force all files created in DISK$USER1:[SIMPSONB] to have no WORLD access, perform the following command:

```
$ SET ACL/OBJECT=FILE/ACL=(DEFAULT_PROTECTION,S:RWED, O:RWED, G:RE, W:) -
      DISK$USER1:[000000]SIMPSONB.DIR
```

Security ACEs

Use *security* ACEs to generate audit events and security alarms whenever users access an object. When a security ACE is triggered and the audit event is written to the security audit file (see Section 17.7), a security alarm is sent (using operator console messenger [OPCOM]) to all enabled security operator consoles. Security ACEs have the following format:

```
(ALARM_JOURNAL=SECURITY, OPTIONS=options , ACCESS=access-mode)
```

To generate an audit event and alarm you must specify one or more of the standard object access modes (READ, WRITE, EXECUTE, DELETE, CONTROL) *and* indicate whether successful (SUCCESS) or failed (FAILURE) accesses are to be audited. For example, to add a security ACE to the SYSUAF file that generates a security alarm whenever a failed write access occurs, perform the following command:

```
$ SET ACL/OBJECT=FILE -
        /ACL=(ALARM_JOURNAL=SECURITY, ACCESS=WRITE+FAILURE) -
        SYS$SYSTEM:SYSUAF.DAT
```

[OpenVMS 6.0]

Beginning with OpenVMS 6.0, the ALARM_JOURNAL ACE has been renamed ALARM and it now generates *only* a security alarm. A new security ACE "AUDIT" has been added that generates *only* an audit record. These new ACEs have the following format:

```
(ALARM=SECURITY, OPTIONS=options , ACCESS=access-mode)  ! Alarm ACE
(AUDIT=SECURITY, OPTIONS=options , ACCESS=access-mode)  ! Audit ACE
```

ACE Options

ACEs can have one or more associated options, which modify the characteristics of the ACE (Table 17.6).

17.4.3 Device Protection

Within the context of system management, device protection primarily controls the ability to initialize, mount, and dismount volumes on a device, and to change device protection attributes. Table 17.7 lists the access required to perform each of these functions.

The PHYSICAL and LOGICAL access types allow a process to perform low-level input-output operations using the queued input-output ($QIO) system service. Most applications that access file-structured devices do not issue direct $QIOs, but rather interface to Digital's Record Management Services (RMS), and thus do not require this level of access. However, physical and logical access is required to modify certain terminal device characteristics, as indicated in Table 17.7.

Table 17.8 compares the device protection defaults and management commands between OpenVMS 5.5 and OpenVMS 6.0.

Table 17.6 Access Control Entry Options

ACE Option	*Function*
DEFAULT	Only valid for identifier and security ACEs associated with directory objects. Specifies that the ACE (less the DEFAULT option) should be applied to all new files created in the directory
HIDDEN	Prevents the ACL from being viewed with the DIR/ACL or DIR/SECURITY commands, unless the subject process has the SECURITY privilege
PROTECTED	Prevents ACEs from being generically deleted with SET ACL/DELETE or EDIT/ACL commands. ACE must be explicitly deleted using SET ACL/ACL=ace/DELETE or SET ACL/DELETE=ALL
NOPROPAGATE	Prevents ACE from being automatically propagated. For example, prevents directory ACEs from propagating to new subdirectories. Similarly, prevents file ACEs from propagating to new file versions
NONE	Specifies that ACE has no options

As you can see from the table, a number of changes occurred between OpenVMS 5.5 and OpenVMS 6.0, including the following:

- Default disk device protection changed to match default tape device protection
- You can establish default protection for disk and tape devices using security templates
- Security templates for terminal devices make the system parameters TTY_PROT and TTY_OWNER obsolete
- SET SECURITY command replaces SET PROTECTION/DEVICE and SET ACL commands

17.4.4 Volume Protection

To access files on a volume, you must first have appropriate access to the volume containing the files. Table 17.9 lists the volume access required to perform various file operations and to change volume protection. To perform file operations on the volume, you must also meet the access requirements of individual files or directories, as discussed in the next section.

For magnetic tape volumes, only READ and WRITE access is meaningful. Tape WRITE access also implies READ access.

Table 17.10 compares the device protection defaults and management commands between OpenVMS 5.5 and OpenVMS 6.0. These defaults can be

Table 17.7 Disk Device Access Requirements

Function	*Required Access*				
OpenVMS 5.0	Read	Write	Physical	Logical	Control
OpenVMS 6.0	Read	Write	Physical	Logical	Control
Perform physical I/O operation, e.g., SET TERM, /SYSPASSWORD, /PERMANENT			•		
Perform logical I/O operation, e.g., SET TERM, /PERMANENT				•	
Mount/dismount device local Either of the following	•	•			•
Mount/dismount device /GROUP GRPNAM and either	•	•			•
Mount/dismount device /SYSTEM SYSNAM and either	•	•			•
Change device protection					•

overridden by using the /PROTECTION and /OWNER qualifiers with the DCL INITIALIZE and MOUNT commands.

17.4.4.1 Volume High-water Marking and Erase_on_delete

You need to consider the following two additional factors in relation to volume protection.

HIGHWATER_MARKING

By default, when you delete a file, disk blocks are deallocated from the file, but the actual data remains intact. High-water marking prevents you from reading data contained in previously deallocated disk blocks. When you create or extend a file, it may map disk blocks that were used by a file that was recently deleted. If high-water making is enabled, OpenVMS records the point at which you write data to the new file and lets you read only up to that point. If you do not require this feature for security purposes, you should disable high-water marking by using the following command:

```
$ SET VOLUME/NOHIGHWATER disk-name
```

The next time you mount the disk, high-water marking is disabled. A system re-boot is required to disable high-water marking on the system disk.

Table 17.8 Device Protection Defaults and Commands

Device Operation	*OpenVMS 5.5*	*OpenVMS 6.0*
Default ownership	[SYSTEM]	[SYSTEM]
Default protection	Disk: (S:RWLP, O:RWLP, G:RWLP, W:RWLP) Tape: (S:RWLP, O:RWLP, G:, W:R) Term: (S:RWLP, O:, G:, W:)	Disk: (S:RWPL, O:RWPL, G:R, W:) Tape: (S:RWPL, O:RWPL, G:R, W:) Term: (S:RWPL, O:RWPL, G:, W:)
Change defaults	Disk and Tape: Hard-coded in OpenVMS Terminal: With TTY_PROT and TTY_OWNER system parameters	SET SECURITY DEVICE.device-type /CLASS=SECURITY /PROTECTION=mask /OWNER=uic /ACL=acl (Where device-type = DISK, TAPE, or TERMINAL)
Change ownership	$ SET PROTECTION/DEVICE /OWNER=uic device	SET SECURITY/OBJECT=DEVICE /OWNER=uic device
Change protection	$ SET PROTECTION/DEVICE /PROTECTION=mask device	SET SECURITY/OBJECT=DEVICE /PROTECTION=mask device
Add ACLs	SET ACL/OBJECT=DEVICE/ACL=acl device	SET SECURITY/OBJECT=DEVICE /ACL=(acl) device

ERASE_ON_DELETE

Erase on delete causes OpenVMS to physically overwrite each block of a file when it is deleted. This prevents someone from reading deallocated disk blocks using special disk scavenging hardware. If this mechanism is not required by your site, disable it as follows:

```
$ SET VOLUME/NOERASE_ON_DELETE disk-name
```

17.4.5 File Object Protection

To access a file, you must first have access to the volume containing the file, as discussed in the pervious section. Next, you need EXECUTE access to all directories between the master file directory (MFD) ([000000]) and the directory that contains the file. Finally, depending on the requested access, you need a combination of access types to the file and the immediate parent of the file (Table 17.11).

Table 17.9 Volume Access Requirements

Volume Operation	*Required Access*				
OpenVMS 5.0	Read	Write	Execute	Delete	Control
OpenVMS 6.0	Read	Write	Create	Delete	Control
Read, list, copy, and print files	•				
Modify or append files	•	•			
Create files	•	•	•		
Delete files	•	•		•	
Change volume protection or ownership					•

For example, the file DISK$USER1:[USER.SIMPSONB]T.TMP has the following associated directory structure:

```
                    ! Owner         Protection

DISK$USER1:         ! [1,4]         (S:RWED,O:RWED,G,W:RWED)  ! Volume Protection

DISK$USER1:[000000] ! [1,4]         (S:RWE,O:RWE,G,W:E)       ! MFD Protection

USER.DIR            ! [1,4]         (S:RWE,O:RWE,G,W:E)       ! Intervening Dir

SIMPSONB.DIR        ! [200,200]     (S:RWE,O:RWE,G,W:RW)      ! Immediate Parent

T.TMP               ! [200,200]     (S:RWED,O:RWED,G,W:RWED)  ! File
```

If user SIMPSONB (UIC [200,200]) wanted to delete the file T.TMP, the user would need the following access:

- DELETE access to the volume DISK$USER1
- EXECUTE access to the MFD, USER.DIR, and SIMPSONB.DIR
- READ access to T.TMP and SIMPSONB.DIR
- WRITE access to SIMPSONB.DIR
- DELETE access to T.TMP

Because all these requirements are met, the user can successfully delete the file.

Table 17.12 compares the file protection defaults and management commands between OpenVMS 5.5 and OpenVMS 6.0.

Table 17.10 Volume Protection Defaults and Commands

Volume Operation	*OpenVMS 5.5*	*OpenVMS 6.0*
Default ownership	UIC of creating process	UIC of creating process
Default protection	Disk and tape: (S:RWED, O:RWED, G:RWED, W:RWED)	Disk and tape: (S:RWCD, O:RWCD, G:RWCD, W:RWCD)
Change defaults	Not supported (hard-coded into OpenVMS)	SET SECURITY VOLUME /CLASS=SECURITY /PROFILE=TEMPLATE /PROTECTION=mask /OWNER=uic /ACL=acl
Change ownership	$ SET VOLUME /OWNER_UIC=uic device To change ownership of a tape volume, you must reinitialize it	SET SECURITY/OBJECT=VOLUME /OWNER=uic device To change ownership of a tape volume, you must reinitialize it
Change protection	$ SET VOLUME /PROTECTION=mask device To change protection of a tape volume, you must reinitialize it	SET SECURITY/OBJECT=VOLUME /PROTECTION=mask device To change protection of a tape volume, you must reinitialize it
Privileges		
BYPASS:	Bypass UIC-based volume protection	
SYSPRV:	Access the volume via the SYSTEM access category	
VOLPRO:	Provides control access to volume	
Add ACLs	Not supported	SET SECURITY/OBJECT= VOLUME/ACL=(acl)

Default File Ownership

By default, when a new *directory* is created, its ownership is established as follows:

1. If the creating process has CONTROL access to the parent directory, the UIC owner of the new directory is set to the UIC of the parent directory
2. Otherwise, the UIC of the new directory is set to the UIC of the creating process

By default, when a new *file* is created, ownership is established as follows:

Table 17.11 File Access Modes

File Operation	*Access Required (F = file, D = parent directory)*[1]				
OpenVMS 5.0 and 6.0	Read	Write	Execute	Delete	Control
List named file			D		
Read, type, print, or view attributes of named file	F		D		
List files using wild-card specification	D				
Read, type, print, or view attributes using wild-card specification	D, F				
Write, modify, append file	D, F	D, F			
Execute named file			D, F		
Create brand-new file	D	D			
Create new version of file	D	D, F		D[2]	
Delete or rename file	D, F	D		F	
Change file attributes	D, F	D, F	F		
Change file ownership and protection	D, F	D, F			F
All operations	D, F	D, F		F	F

[1]You also must have EXECUTE access to all intervening directories.

[2]DELETE is required only when creating a new file would cause a version of the same file to be deleted (due to file version limits)

1. When a new version of an existing file is created, it is assigned the UIC owner of the previous version of the file only if the creating process has CONTROL access to the file
2. When a new file is created, it is assigned the UIC owner of the parent directory only if the creating process has CONTROL access to the parent directory
3. If the creating process does not have CONTROL access, the UIC of the created file is set to the UIC of creating process.

Default File Protection

When you create a directory, its protection is established from the parent directory. When you create a directory in the MFD (e.g., DISK-$USER1:[USER]), the protection for the directory is established from the MFD [000000]. By default, the protection of the MFD is as follows:

```
(S:RWE, O:RWE, G:RE, W:E)
```

Table 17.12 File Protection Defaults and Commands

File Operation	*OpenVMS 5.5*	*OpenVMS 6.0*
Default ownership	Usually inherits UIC of creating process	Usually inherits UIC of creating process
Default protection	(S:RWED, O:RWED, G:RE, W:)	(S:RWED, O:RWED, G:RE, W:)
Change defaults	RMS_FILEPROT system parameter or SET PROTECTION=mask/DEFAULT or Add DEFAULT_PROTECTION ACE	Same as OpenVMS 5.5
Change ownership	SET FILE/OWNER=uic	SET SECURITY/OBJECT=FILE /OWNER=uic file
Change protection	SET PROTECTION=mask file SET FILE/PROTECTION=mask file	SET SECURITY/OBJECT=FILE /PROTECTION=mask file
Display security attributes	DIR/OWNER file DIR/PROTECTION file DIR/ACL file DIR/SECURITY file	DIR/OWNER file DIR/PROTECTION file DIR/ACL file DIR/SECURITY file SHOW SECURITY file
Privileges:		
SYSPRV:	Allows access via the SYSTEM access category, regardless of subject's UIC	
GRPPRV:	Allows access via the SYSTEM access category if the subject's UIC group matches the object's UIC group	
READALL:	Allows READ access, regardless of subject's UIC	
BYPASS:	Allows subject to completely bypass UIC-based protection	
Add ACLs	SET ACL/OBJECT= FILE/ACL=acl file	SET SECURITY/ACL=(acl) file

When you create a *new file*, its protection is set to the default protection of the creating process. The default protection of a process is established during system login with the following command:

```
$ SET PROTECTION=(S:rwed,O:rwed,G:rwed,W:rwed)/DEFAULT
```

If you don't explicitly set default protection with the SET PROTECTION command, default protection is set to the value specified by the system parameter RMS_FILEPROT. The default value for RMS_FILEPROT (64000 decimal) translates to a protection mask as follows:

Table 17.13 Queue and Job Access Requirements

Queue Operation	*Required Access (Q=queue, J=job)*				
OpenVMS 5.0	Read	Write	Execute	Delete	Control
OpenVMS 6.0	Read	Submit	Manage	Delete	Control
Submit jobs to queue		Q			
Control queue			Q		
View job information	J		Q		
Delete a job			Q	J	

```
   F        A        0        0      ! Hexadecimal for 64000
 1111     1010     0000     0000     ! Convert to binary
 0000     0000     0101     1111     ! Invert bits from left to right
(S:RWED, O:RWED, G:R E , W:     )    ! 0 = Set, 1 = Not Set
```

When you create a new version of an *existing file*, its default protection is set to the protection of the previous version of the file.

17.4.6 Queue Object Protection

Table 17.13 details the access modes required to perform various queue and job operations.

Table 17.14 summarizes the queue protection defaults and DCL command differences between OpenVMS 5.0 and 6.0.

17.5 Security Class Object Protection

[OpenVMS 6.0]

To modify default object security templates using the commands listed in the previous sections (e.g., SET SECURITY object/CLASS=SECURITY), you need appropriate access to the security class object as indicated in Table 17.15.

By default, the security class object has the following protection:

```
$ SHOW SECURITY/CLASS=SECURITY SECURITY_CLASS

SECURITY_CLASS object of class SECURITY_CLASS
     Owner: [SYSTEM]
     Protection: (System: RW, Owner: RW, Group: R, World: R)
     Access Control List: <empty>
```

Table 17.14 Queue Protection Defaults and Commands

Queue Operation	*OpenVMS 5.5*	*OpenVMS 6.0*
Default ownership	[SYSTEM]	[SYSTEM]
Default protection	(S:E, O:D, G:R, W:W)	(S:M, O:D, G:R, W:S)
Change defaults	N/A	SET SECURITY QUEUE /CLASS=SECURITY /PROTECTION=mask /OWNER=uic /ACL=acl
Change ownership	$ SET QUEUE /OWNER_UIC=uic queue $ INITIALIZE/QUEUE /OWNER_UIC=uic queue	SET SECURITY/OBJECT=QUEUE /OWNER=uic queue
Change protection	$ SET QUEUE /PROTECTION=mask queue $ INIT/QUEUE /PROTECTION=mask	SET SECURITY/OBJECT=QUEUE /PROTECTION=mask queue
Privileges:		
OPER:	Provides EXECUTE access to all queues	
SYSNAM:	Stops and starts the queue manager	
Add ACLs	$ SET ACL/OBJECT=QUEUE /ACL=acl queue	SET SECURITY/OBJECT=VOLUME /ACL=(acl) queue

17.6 Protected Subsystems

[OpenVMS 6.0]

Sometimes users need elevated access when they execute a particular application (e.g., to access associated data files). One solution is to create a general identifier, grant it to selected users, and add ACLs to the necessary data files granting access to identifier holders. This strategy allows access from the application; however, there is no mechanism to prevent savvy users from manipulating data files directly from DCL.

Another option is to install an executable image with the elevated privileges using the Install utility, as follows:

```
$ INSTALL:==$INSTALL/COMMAND_MODE
$ INSTALL ADD DISK$TOOLS:TASM.EXE/PRIVILEGE=SYSPRV
```

Table 17.15 Security Class Access Requirements

Security Operation	*Required Access*		
	Read	Write	Control
Display a security template	•		
Modify a security template		•	
Change the protection of the security class object			•

This technique does temporarily elevate users' access, but enabling powerful privileges such as SYSPRV creates additional security exposure.

To resolve these problems, OpenVMS 6.0 includes a *protected subsystem* mechanism that effectively allows you to associate an identifier (rather than a privilege) with an executable image. When a user activates an image that has an associated subsystem identifier, the identifier is temporarily enabled for the user's process. When the user exits the application, the identifier is disabled. By applying ACLs to subsystem objects, you can allow users to access objects only when they are executing within the subsystem.

To use protected subsystems, you must enable the processing of subsystem ACLs on a volume by volume basis. By default, the system disk is automatically enabled. To enable additional disks mount them during system startup with the /SUBSYSTEM qualifier, as follows:

```
$ MOUNT/CLUSTER/SUBSYSTEM $1$DUA1 USER1
$ SET VOLUME/SUBSYSTEM $1$DUA1      ! If device is already mounted
```

To demonstrate the use of protected subsystems, consider an application called XYZ that consists of an executable image XYZ.EXE and a number of data files XYZ1.DAT, XYZ2.DAT, etc. Assume that when users are executing XYZ.EXE they should have full access to the associated data files.

The first step is to create a subsystem identifier with the Authorize utility, as follows:

```
UAF> ADD/ID/ATTRIBUTES=(SUBSYSTEM, RESOURCE) SYSTEM_XYZ
```

Grant the identifier to yourself or to the person who will be configuring subsystem objects. This is necessary because to add subsystem ACLs, you must hold the identifier.

```
UAF> GRANT/ID/ATTRIBUTES=(SUBSYSTEM, RESOURCE) SYSTEM
```

Then, add a subsystem ACL to XYZ.EXE

```
$ SET SECURITY/ACL=(SUBSYSTEM,IDENTIFIER=SYSTEM_XYZ, -
    ATTRIBUTES=RESOURCE) XYZ.EXE
```

Next, add normal ACLs to the database files that allow access to holders of the SYSTEM_XYZ identifier

```
$ SET SECURITY/ACL=(IDENTIFIER=SYSTEM_XYZ, ACCESS=READ+WRITE) XYZ*.DAT
```

To prevent access from any users outside the subsystem, you should remove UIC-based group and world access, make SYSTEM_XYZ the owner, and add an identifier to deny access to users who have not acquired the SYSTEM_XYZ subsystem identifier

```
$ SET SECURITY/PROTECTION=(G,W)/OWNER=SYSTEM_XYZ-
/ACL=(IDENTIFIER=*, ACCESS=NONE)-
/AFTER=(IDENTIFIER=SYSTEM_XYZ,ACCESS=READ+WRITE) XYZ*.DAT
                                              ! Place at end of ACL

$ DIR/SECURITY XYZ1.DAT

Directory SYS$SYSROOT:[SYSMGR]

XYZ1.DAT;1      SYSTEM_XYZ        (RWED,RWED,,)
  (IDENTIFIER=SYSTEM_XYZ,ACCESS=READ+WRITE)        ! Allow access
  (IDENTIFIER=*,ACCESS=NONE)                       ! Deny to all others

Total of 1 file.
```

Now anyone who executes XYZ.EXE will temporarily acquire the SYSTEM_XYZ identifier, allowing him or her to read and write to the XYZ data files. This is demonstrated by the following:

```
$ SPAWN/NOWAIT/OUT=T.TMP RUN XYZ    ! Spawn subprocess to invoke XYZ
%DCL-S-SPAWNED, process SYSTEM_1 spawned

$ SHOW PROCESS/RIGHTS SYSTEM_1      ! View the rights of subprocess

26-MAY-1995 14:20:16.12  User: SYSTEM    Process ID: 00000052
                         Node: BIRCH     Process name: "SYSTEM_1"

Process rights:
```

```
 INTERACTIVE
 LOCAL

Subsystem rights:
 SYSTEM_XYZ                          ! <=== SUBSYSTEM rights ARE enabled

System rights:
 SYS$NODE_BIRCH
```

17.7 Security-related System Parameters

Several system parameters (each prefixed with LGI_) control a variety of OpenVMS security mechanisms. To modify these parameters use AUTOGEN as discussed in chapter 18.

```
LGI_RETRY_LIM          ! Default is 3 attempts
```

This procedure specifies the maximum number of failures that can occur during a dialup, network, or terminal server login. If the limit is exceeded, the current connection is terminated, requiring the user to re-dial, re-issue a SET HOST, or re-connect (from terminal servers).

The purpose of LGI_RETRY_LIM is to increase the amount of time it takes to repetitively guess passwords. By default, a person guessing passwords via a dialup line would have to re-dial after every three failed login attempts.

```
LGI_RETRY_TMO          ! Default is 20 seconds
```

This procedure specifies the login time-out period. By default, if after receiving the username or password prompt, you do not respond within 20 seconds a login time-out failure occurs.

Once you receive a login prompt, OpenVMS has already created a process to handle the anticipated login. LGI_RETRY_TMO helps conserve system resources by terminating a login (and deleting any associated processes) if the login is not completed within a reasonable amount of time.

```
LGI_PWD_TMO            ! Default is 30 seconds
```

This procedure indicates the amount of time users have to enter the system password (if enabled) when performing a login. If the time-out period is exceeded a login time-out failure occurs.

```
LGI_BRK_LIM            ! Default is 5 failures
```

This procedure specifies the maximum number of login failures that can occur from a specific source before the failures are considered a break-in attempt. When the first login failure occurs, it is set to expire in LGI_BRK_TMO seconds. If additional login failures occur before the expiration time, the expiration time is extended to the current time plus LGI_BRK_TMO seconds. By default, if five or more failures occur before the count expires, OpenVMS assumes a break-in attempt is in progress and begins evasive action.

```
LGI_BRK_TMO          ! Default is 300 seconds (5 minutes)
```

(See LGI_BRK_LIM.)

```
LGI_BRK_DISUSER      ! Default is FALSE (0)
```

If this parameter is set to 1, and a break-in attempt occurs, the user account named as the source of the attempt is disabled by setting the account's DISUSER login flag. To re-activate the account, you must manually remove the DISUSER flag with the Authorize utility.

By default, LGI_BRK_DISUSER is *not* set. Therefore, accounts are not automatically disabled when they are involved in break-in attempts. If you set this parameter to 1, break-in evasion is more effective. However, you run the risk that all accounts will be disabled by a person generating login failures to every valid username.

```
LGI_HID_TIM          ! Default is 300 seconds
```

Once a break-in has been detected, OpenVMS initiates evasive action for the period of time specified by LGI_HID_TIM. During the evasion period, OpenVMS rejects even valid usernames and passwords from the account suspected in the break-in.

```
LGI_BRK_TERM         ! Default is (1)
```

When set to 1, login failures by the same username, but from multiple terminals are counted separately. The default setting is acceptable if logins are initiated only from direct terminal ports (TT or TX devices). However, if users log in from terminal servers, you should seriously consider setting LGI_BRK_TERM to zero so that login failures are tracked by username only.

For example, if LGI_BRK_TERM is set to 1, and a single user generates nine login failures from a terminal server, the failures are counted as follows:

```
$ SHOW INTRUSION

Intrusion       Type       Count      Expiration      Source
   TERM_USER    SUSPECT      3        14:43:50.22     _LTA160:SIMPSONB
   TERM_USER    SUSPECT      3        14:44:19.72     _LTA158:SIMPSONB
   TERM_USER    SUSPECT      3        14:44:32.62     _LTA159:SIMPSONB
```

After every three login failures (LGI_RETRY_LIM), the terminal session is terminated. When the user re-connects, the terminal name changes from LTA159: to LTA160:. Because the username and terminal are tracked separately, OpenVMS does not combine the total login failures. Therefore, LGI_BRK_LIM is not exceeded and the login failures are considered as SUSPECT rather than a bona fide break-in.

When LGI_BRK_TERM is set to zero, OpenVMS tracks login failures by username only. The same nine login failures would be counted as follows:

```
$ SHOW INTRUSION

Intrusion       Type       Count      Expiration      Source
   TERM_USER    INTRUDER     6        13:05:36.61     _OPA0:SIMPSONB
```

Note that the intrusion type is INTRUDER, indicating that a break-in has been detected and evasion action is in progress. Furthur logins from that source are disabled until the intrusion record expires. As an alternate, you can delete an intrusion record manually, as follows:

```
$ DELETE/INTRUSION _OPA0:SIMPSONB        ! Spell the source EXACTLY
$ SHOW INTRUSION
%SHOW-F-NOINTRUDERS, no intrusion records match specification
```

17.8 Network Security

When your systems are part of a DECnet network, or if you use terminal servers, you should carefully consider the following network security issues:

- Terminal server passwords
- Default DECnet access
- DECnet proxy access

17.8.1 Terminal Server Passwords

Most terminal servers have an associated *server password* and a *privilege password.*

Privilege Password

The terminal server privilege password controls access to server management functions such as testing ports and changing the server configuration.

By default the privilege password is set to SYSTEM. To prevent malicious users from changing your server configurations, you should establish new privilege passwords as follows:

```
Local> SET PRIVILEGE
Password: old-password
Local> SET SERVER PRIVILEGE PASSWORD new-password
Local> DEFINE SERVER PRIVILEGE PASSWORD new-password
```

Server Password

The server password works much like the OpenVMS system password in that it is enabled on a port by port basis. If users attempt to connect to a port when a server password is enabled, they receive the # prompt and must enter the server password before they can continue.

The factory default server password is ACCESS. You should take care to change this password and enable server passwords on dialup ports (at a minimum). Change the server password as follows:

```
Local> SET PRIVILEGE
Password: privilege-password
Local> SET SERVER PASSWORD password
Local> DEFINE SERVER PASSWORD password
```

To enable a port to require a server password perform the following procedure:

```
Local> SET PRIVILEGE
Password:privilege-password

Local> DEFINE PORT n PASSWORD ENABLED
Local> LOGOUT PORT n
```

17.8.2 Default DECnet Access

When you initially configured DECnet on your system, you may have established a default DECnet account. The purpose of this account is to allow remote DECnet users minimal access to your system *without* requiring standard username and password authentication.

To verify if you have a default DECnet account, execute the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP> SHOW EXECUTOR CHARACTERISTICS
.
.
Nonprivileged user id     = DECNET
Nonprivileged password    = FOOBAR
Default access            = incoming and outgoing
.
.
```

You must be enabled for BYPASS privilege to see the nonprivileged password. If something like the foregoing is displayed, and the user id specified exists in your UAF, a default DECnet account is enabled.

If you have a default DECnet account, ensure that its UAF parameters are set as follows:

- Does not have system UIC
- Has only TMPMBX and NETMBX privileges
- Is restricted to network access only

At a minimum, users from remote nodes can copy files to and from the default login directory of the DECnet account. For example:

```
[on node LOTUS, from a nonprivileged user account]

$ COPY T.TMP BIRCH::
$ DIR BIRCH::

Directory of BIRCH::SYS$SPECIFIC:[DECNET]

T.TMP;1

Total of 1 file.
```

A potential problem with default DECnet accounts is that remote users can copy large files to your system and cause disk space to be exhausted.

The biggest concern, however, is that remote users have access to all files on your system via the WORLD access category. If this is a concern, you should restrict access to the WORLD category or disable default DECnet access.

17.8.3 DECnet Proxy Access

DECnet proxy access provides another means of facilitating access by remote users. A proxy allows a remote user access to the local system without having to specify a username and password.

The most common use of proxies is when a user has several user accounts on different DECnet nodes. Without proxies, the user would have to specify a username and password when transferring files between systems. Assume user SIMPSONL has accounts on node BIRCH and node LOTUS and wants to copy a file from LOTUS:: to BIRCH::. This would require the following command:

```
[on node LOTUS::]
$ COPY T.TMP BIRCH"SIMPSONL password"::
```

To eliminate the need to add the username and password in the COPY command, you should create a proxy entry on node BIRCH::, as follows:

```
$ UAF
UAF> ADD/PROXY LOTUS::SIMPSONL SIMPSONL/DEFAULT
```

The foregoing proxy allows any remote DECnet requests originating from node LOTUS and user SIMPSONL to use the local user account SIMPSONL. The /DEFAULT option specifies that the proxy entry be used to determine default access by the specified user, thus superseding the default DECnet account (if one exists). The copy could then be performed as follows:

```
[on node LOTUS::]
$ COPY T.TMP BIRCH::
```

Proxy entries should never allow access to privileged accounts. Otherwise, malicious users on remote nodes outside your control could gain full access to your system.

17.9 OpenVMS Security Auditing

Robust implementation of the OpenVMS security mechanisms discussed within this chapter are only the first step. You should also perform regular system auditing to ensure that security mechanisms are effective.

During system startup, OpenVMS creates the audit server process (AUDIT_SERVER). When active, the audit server logs selected security events by sending security broadcast messages (via OPCOM) and by writing audit records to the *audit log file.* By default, the audit login file consists of the following file:

```
SYS$MANAGER:SECURITY_AUDIT.AUDIT$JOURNAL
```

[OpenVMS 6.0]

Beginning with OpenVMS 6.0, the audit journal file is named as follows:

```
SYS$MANAGER:SECURITY.AUDIT$JOURNAL
```

Depending on your security requirements, you can enable auditing for a variety of events, as listed in Table 17.16.

To list the events that are enabled on your system, issue the following command:

```
$ SHOW AUDIT

System security alarms currently enabled for:
  ACL
  Authorization
  Breakin: dialup,local,remote,network,detached
  Logfailure: batch,dialup,local,remote,network,subprocess,detached

System security audits currently enabled for:
  ACL
  Authorization
  Breakin: dialup,local,remote,network,detached
  Logfailure: batch,dialup,local,remote,network,subprocess,detached
```

With versions before OpenVMS 6.0, events can be enabled only to generate a security alarm that when triggered generates a security broadcast *and* a written record to the audit log file. Beginning with OpenVMS 6.0, if an event is enabled for an alarm, it generates only a security broadcast. To generate an

Table 17.16 Auditable OpenVMS Events

Audited Event	*Audit Option*	*OpenVMS 5.5*	*OpenVMS 6.0*
General object access	ACCESS	No	Yes
Objects with alarm ACEs	ACL	Yes	Yes
Audit modifications	AUDIT	Yes	Yes
Authorization database modified (SYSUAF, RIGHTSLIST, NETPROXY)	AUTHORIZATION	Yes	Yes
Break-in detected	BREAKIN	Yes	Yes
Connection from DECnet, DECwindows, $IPC, or SYSMAN	CONNECTION	No	Yes
Object creation	CREATE	No	Yes
Object deaccess	DEACCESS	No	Yes
Object deletion	DELETE	No	Yes
All specified file access	FILE_ACCESS	Yes	Use ACCESS /CLASS=FILE
Use of identifier	IDENTIFIER	No	Yes
Installed image added or removed	INSTALL	Yes	Yes
Login failures	LOGFAILURE	Yes	Yes
Specified logins	LOGIN	Yes	Yes
Specified logouts	LOGOUT	Yes	Yes
Device mounts	MOUNT	Yes	Yes
Modification of NCP (DECnet) database	NCP	No	Yes
Use of privilege	PRIVILEGE	No	Yes
Process control functions (e.g., create process, delete process)	PROCESS	No	Yes
System parameter modification	SYSGEN	No	Yes
System time modification	TIME	No	Yes

entry in the audit log file you must also enable the event for audits. To enable the alarm or audit events, use the following commands:

```
$ SET AUDIT/ALARM/ENABLE=event      ! OpenVMS 6.0 :Alarm only
                                    ! OpenVMS 5.5: Alarm AND Audit

$ SET AUDIT/AUDIT/ENABLE=event      ! OpenVMS 6.0: Audit only
```

You should choose the events you audit carefully and select only those you really need. Excessive auditing can easily generate thousands of audit events per minute (or more), which will seriously degrade system performance. By

default, only the operator console is enabled to receive security alarms. To receive alarms at other terminals, you should add the following command to your login procedure:

```
$ REPLY/ENABLE=SECURITY          ! Requires OPER and SECURITY privilege
```

17.9.1 Generating Audit Reports

Once you enable the events you want to audit, you can generate reports using the ANALYZE/AUDIT facility. On a daily basis you should generate a summary report that indicates the number of audit records generated for each event type, as follows:

```
$ ANALYZE/AUDIT/SUMMARY/SINCE=TODAY/OUT=SUMMARY.RPT SYS$MANAGER
$ TYPE SUMMARY.RPT
Total records read:       263       Records selected:          8
Record buffer size:       917
Successful logins:          0       Object creates:            0
Successful logouts:         0       Object accesses:           0
Login failures:             0       Object deaccesses:         0
Breakin attempts:           0       Object deletes:            0
System UAF changes:         0       Volume (dis)mounts:        0
Rights db changes:          3       System time changes:       0
Netproxy changes:           0       Server messages:           0
Audit changes:              5       Connections:               0
Installed db changes:       0       Process control audits:    0
Sysgen changes:             0       Privilege audits:          0
NCP command lines:          0
```

If you see any records for UAF, audit, installed images, SYSGEN, or NCP (network control program) changes you should investigate them further, unless you can account for them. Login failures and breakin attempts are more difficult to analyze, because if you have a large user population you'll receive many such audit events each day. To deal with these events you should record the typical number of login failures and breakin attempts that you normally receive and explore further if record counts exceed these norms.

To get a little more information, you can generate a brief audit report, as follows:

```
$ ANALYZE/AUDIT/BRIEF/SINCE=TODAY/OUT=BRIEF.RPT SYS$MANAGER
$ TYPE BRIEF.RPT
```

```
Date/Time                 Type      Subtype          Node   Username  ID        Term
------------------------------------------------------------------------------
26-MAY-1995 12:14:52.99   AUDIT     AUDIT_TERMINATE  BIRCH            0000004D
26-MAY-1995 14:12:34.16   RIGHTSDB  RDB_ADD_ID       BIRCH  SYSTEM    0000004B  OPA0:
26-MAY-1995 14:13:53.25   RIGHTSDB  RDB_MOD_ID       BIRCH  SYSTEM    0000004B  OPA0:
26-MAY-1995 15:47:09.90   RIGHTSDB  RDB_ADD_ID       BIRCH  SYSTEM    0000004B  OPA0:
```

In analyzing the above report, you should look for suspect events, such as

- Many login failures from the same source
- Excessive login attempts during evenings or weekends
- Any changes by unfamiliar usernames
- Newly installed images, especially if added with privileges

If you see any changes that look suspect, for example the RIGHTSDB changes in the foregoing listing, you can produce a full report on just those events by using the /EVENT qualifier, as follows:

```
$ ANALYZE/AUDIT/FULL/SINCE=TODAY/EVENT_TYPE=RIGHTSDB/OUT=FULL.RPT -
      SYS$MANAGER
$ TYPE FULL.RPT

Security alarm (SECURITY) and security audit (SECURITY) on BIRCH,
system id:          1025
Auditable event:    Identifier added
Event time:         26-MAY-1995 14:12:34.16
PID:                0000004B
Process name:       SYSTEM
Username:           SYSTEM
Process owner:      [SYSTEM]
Terminal name:      OPA0:
Image name:         BIRCH$DUA1:[SYS0.SYSCOMMON.][SYSEXE]AUTHORIZE.EXE
Identifier name:    TASM_SYSTEM
Identifier value:   %X80010003
Attributes:         SUBSYSTEM

Security alarm (SECURITY) and security audit (SECURITY) on BIRCH,
system id:          1025
Auditable event:    Identifier modified
Event time:         26-MAY-1995 14:13:53.25
PID:                0000004B
Process name:       SYSTEM
Username:           SYSTEM
Process owner:      [SYSTEM]
Terminal name:      OPA0:
```

```
Image name:          BIRCH$DUA1:[SYS0.SYSCOMMON.][SYSEXE]AUTHORIZE.EXE
Identifier name:     TASM_SYSTEM
Identifier value:    %X80010003
Attributes:          SUBSYSTEM
New attributes:      DYNAMIC,SUBSYSTEM

Security alarm (SECURITY) and security audit (SECURITY) on BIRCH,
system id:           1025
Auditable event:     Identifier added
Event time:          26-MAY-1995 15:47:09.90
PID:                 0000004B
Process name:        SYSTEM
Username:            SYSTEM
Process owner:       [SYSTEM]
Terminal name:       OPA0:
Image name:          BIRCH$DUA1:[SYS0.SYSCOMMON.][SYSEXE]AUTHORIZE.EXE
Identifier name:     SYSTEM_XYZ
Identifier value:    %X80010004
Attributes:          RESOURCE,SUBSYSTEM
```

As the foregoing report indicates, two new identifiers (TASM_SYSTEM and SYSTEM_XYZ) were added by someone logged into the SYSTEM account on the console terminal (OPA0:). If audit reports indicate changes with which you are not familiar you should investigate further.

Once you have created a set of standard reports, you should assemble them into a command procedure and submit the procedure as an automated batch job, as discussed in chapter 12.

17.10 Additional References

[OpenVMS 5.0]

Guide to OpenVMS System Security, Digital, June 1989.
OpenVMS Access Control List Editor Manual, Digital, April 1988.
OpenVMS Networking Manual, Digital, April 1988.

[OpenVMS 6.0]

OpenVMS System Manager's Manual: Essentials, chapter 11, "Security Considerations," Digital, March 1994.
OpenVMS System Manager's Utility Manual, ANALYZE/AUDIT, Digital, March 1994.
OpenVMS VAX Guide to System Security, Digital, March 1994.

18 PERFORMANCE MANAGEMENT

18.1 Chapter Objectives

- Achieve performance requirements dictated by service-level objectives
- Monitor and analyze application performance
- Identify and eliminate resource bottlenecks
- Adjust system parameters using SYSMAN and AUTOGEN

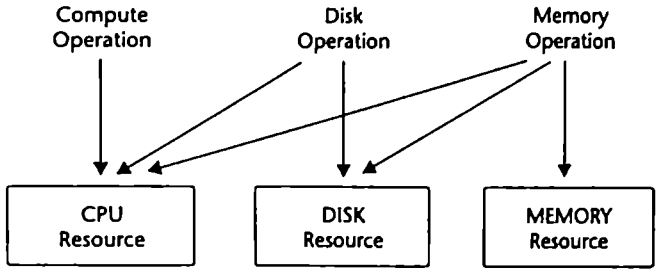
18.2 Introduction

An OpenVMS system consists of three major resources: CPU, memory, and disk. Every application uses each of these resources to varying degrees. For example, a statistical analysis application might rely mostly on the CPU and less on memory and disk. On the other hand, a database update might perform many disk operations while requiring only moderate amounts of CPU and memory resources.

CPU, memory, and disk resources are interrelated. Disk I/O affects a physical disk, but it also requires CPU time to set up and complete the request. Similarly, requests for pages of memory require CPU time and may require disk I/O if the requested pages are no longer located in physical memory. Figure 18.1 illustrates how various application requests affect each of the major system resources.

Some performance management approaches focus solely on monitoring each system resource, detecting when they become saturated, and tuning or load-balancing them until utilization seems acceptable. This approach relies on the assumption that as long a resource is not overutilized, user application response time will be acceptable. Unfortunately, this is not necessarily the

Figure 18.1 System Resource Dependencies



case. For example, an application might issue I/O requests to a dedicated disk at a rate less than 10% of the capacity of the drive. However, just because a drive is underutilized doesn't mean its response time is acceptable. Similarly, it certainly is not a good sign if the CPU of your system is always 85% busy, but it's not that important if application response times are consistently twice as fast as service-level objectives. Users don't care how well the system performs; they only care how well their applications perform. This type of objective-based methodology is illustrated in Figure 18.2 and forms the basis of the approach detailed in this chapter.

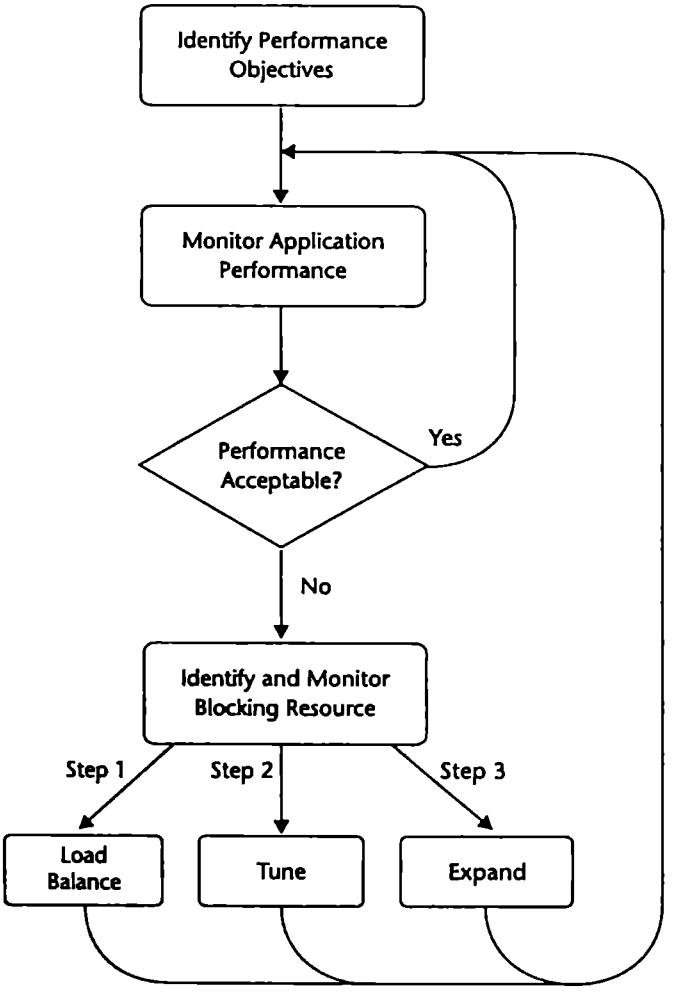
With this approach you start by identifying performance objectives for each application in terms of response time. Then you consistently monitor application performance to ensure that objectives are met. If response times become unacceptable, focus on the system resource on which the application is most dependent. Finally, attempt to load balance, tune, and expand (in that order) the bottlenecked resource until response time returns to an acceptable level. Each of these steps is detailed in the sections that follow.

18.3 Identifying Performance Objectives

Objectives are the most important input to the performance management process. Before you can begin to address or even identify a problem, you must understand exactly what level of performance is considered acceptable. Reasonable objectives should be developed by consulting with users and management. These objectives guide your performance management activities and assist you in reconciling performance trade-offs between one application and another.

Performance objectives should indicate how quickly the system should process and respond to particular user requests or activities. This can be diffi-

Figure 18.2 Objective-based Performance Management



cult to specify for applications that involve many different functions, but you don't necessarily have to establish and track response times for every activity. To approximate overall response time, focus on key activities or activities that seem to be representative of overall use. For example, for a word-processing application, you might want to measure the time required to invoke the application, or the time to perform a spell-check on an average-sized document. On the other hand, for a batch job that performs a database update, you

Table 18.1 Sample Application Performance Objectives

Application	*Average Uses*	*Activity*	*Response Time*	*Consistency*
Text processing utility (TPU)	20	Start TPU editor	00:00:10	95%
Login	100	Get DCL prompt	00:00:15	95%
WORDPERFECT	30	Spell-check 4K document	00:01:30	95%
DB_UPD	1	Database update batch job	01:30:00	95%

might want to measure only the total elapsed time it takes to perform the update.

Application objectives should also include an estimated number of concurrent uses. By documenting this estimate, you'll be better prepared to explain and deal with deteriorating performance when the number of users accessing an application mysteriously increases from 30 to 60. Finally, no OpenVMS system can produce a particular response time 100% of the time, because of unpredictable events such as VMScluster state transitions and hardware errors. To weed out these extremes, you should decide on a consistency factor that specifies the percentage of time you expect to meet response time objectives. For example, if you agree on a 95% consistency factor, discard the worst 5% of the response times you sample. Once you agree on performance objectives, formalize them in writing, as illustrated in Table 18.1.

18.4 Monitoring Application Performance

After you've identified performance objectives, you should regularly monitor whether application performance goals are being satisfied. On a weekly or monthly basis you should summarize response times and publish the results to users or management. Such a report also becomes a historical record of response times, which is immensely helpful in analyzing performance over a longer term.

18.4.1 Monitoring Batch Jobs

You can monitor resource usage and response times for batch jobs simply by reviewing the resource accounting information included at the end of the log file for each batch job. To ensure that this information is retained, include the /KEEP and/LOG qualifiers when jobs are submitted. By default, a log file is

created in the job owner's default directory. The log file is given the same name as the command procedure that submitted it, except with a file extension of .LOG. For example:

```
$ SUBMIT/KEEP/NOPRINT DB_UPD.COM
.
$ TYPE DB_UPD.LOG
.
.
.
SYSTEM     job terminated at 20-SEP-1995 10:28:10.17

Accounting information:
Buffered I/O count:           1545    Peak working set size:       912
Direct I/O count:             1034    Peak page file size:       11134
Page faults:                  8512    Mounted Volumes:               1
Charged CPU time:    0 00:20:00.46    Elapsed time:      0 01:04:25.59
```

When you review the log file, you should record the following information:

- Direct I/O count: Usually an indication of disk I/O activity
- Page faults: Number of requests for pages of memory
- Charged CPU time: Total amount of CPU time used
- Elapsed time: Total time to complete job

The procedures for analyzing these statistics are discussed later in this chapter.

18.4.2 Monitor Interactive Jobs

Measuring interactive activity is difficult because the system does not normally record resource usage statistics until a user logs out. You can gather more timely statistics using one or more of the following techniques.

SHOW PROCESS

You can monitor the resource utilization of an active process by using the following command:

```
$ SHOW PROCESS/CONTINUOUS/ID=process-id

$ SHOW PROCESS/CONTINUOUS/ID=28

Process AUDIT_SERVER      17:58:30
```

```
State                   HIB       Working set              651
Cur/base priority      10/8       Virtual pages           3207
Current PC          7FFEDF8A       CPU time      000:00:00:03.34
Current PSL         03C00000       Direct I/O               75
Current user SP     7FF8C8F8       Buffered I/O             39
PID                 00000028       Page faults             504
UIC                 [SYSTEM]       Event faults         E0000001
                                                        80000000

BIRCH$DUA1 : [SYS0.SYSCOMMON.] [SYSEXE]AUDIT_SERVER.EXE;1
```

After you issue the command, resource usage statistics are displayed and continuously updated every few seconds. You can get a general idea of how much CPU, memory, and disk resources the process is using by watching the CPU time, page faults, and direct I/O statistics. However, because statistics are constantly updated, it is difficult to collect precise data with this method.

ACCOUNTING Utility

By periodically enabling *image-level accounting* you can force the system to record resource utilization statistics after every image termination. Enable image-level accounting using the following command:

```
$ SET ACCOUNTING/ENABLE=IMAGE
```

CAUTION: If your normal system workload involves frequent image activations, the overhead of image-level accounting may substantially degrade system performance. Therefore, you should enable image accounting for only brief periods of time.

To disable image accounting use the following command:

```
$ SET ACCOUNTING/DISABLE=IMAGE
```

Once you have collected image accounting data, you can produce a summary report using the command following this paragraph. The commands assume that your default directory is set to SYS$MANAGER or that you have defined a system-wide logical (ACCOUNTNG) that specifies the location of your accounting data file, as discussed in chapter 5. The following report provides the same basic data as SHOW PROCESS, except you can also determine the percentage of page faults that require a disk I/O (a hard fault) by referring to the Page I/O Reads statistics. The procedures for analyzing these statistics are discussed later in this chapter.

```
$ ACCOUNTING /TYPE=IMAGE/SUMMARY=(IMAGE)/OUTPUT=IMAGE.RPT-
_$  /REPORT=(RECORDS,PROCESSOR,ELAPSED,FAULTS,PAGE_READS,DIRECT_IO) ACCOUNTING
$ TYPE IMAGE.RPT
From: 7-JUN-1995 12:50 VAX/VMS Accounting Report To: 7-JUN-1995 12:52

Image       Total       Elapsed          Page            Page I/O  Direct  I/O
Name        Processor   Time             Time            Faults    Reads
            Records
-------------------------------------------------------------------------------
ACC             4     0 00:00:01.86    0 00:00:06.74      260       63      19
DELETE          1     0 00:00:00.28    0 00:00:00.81       59        5       5
DIRECTORY       2     0 00:00:00.48    0 00:00:01.07      127        4       5
HARD            6     0 00:00:00.13    0 00:00:00.19      140        6       0
SET             2     0 00:00:00.08    0 00:00:00.73       51        4       0
SHOW            1     0 00:00:00.17    0 00:00:00.38       50        2       0
TPU             4     0 00:00:10.07    0 00:00:16.77     6836       13      28
```

Get Job/Process Information Function (F$GETJPI)

To measure the performance of a specific activity without enabling image-level accounting, you can use the lexical function Get Job/Process Information (F$GETJPI). The following DCL command procedure (BENCH.COM) uses F$GETJPI to gather usage statistics before and after a specific function is performed. BENCH then calculates total elapsed time and resource utilization for the activity.

```
$ ! BENCH.COM
$ !
$ ! Program to benchmark a system activity
$ !
$ ! 19-Sep-92 L. Baldwin
$ !
$ SET NOVERIFY
$et:==edit/tpu/nojournal/read/noinit
$LOOP:
$ min=f$cvtime(,,"MINUTE")
$ sec=f$cvtime(,,"SECOND")
$ hun=f$cvtime(,,"HUNDREDTH")
$ start_et=hun + (sec*100) + (min*60*100)      ! Get elapsed time (in 1/100ths)

$ start_dio=f$getjpi(0,"DIRIO")                ! Direct (disk) I/O
$ start_pf=f$getjpi(0,"PAGEFLTS")              ! Page Faults
$ start_cpu=f$getjpi(0,"CPUTIM")               ! CPU time (in 1/100ths)
$ run sys$manager:hard                         ! Get Hard page faults
$ start_hard = f$integer(hardflts)             ! Hard sets HARDFLTS symbol
$ !
```

```
$ ! Add system activity here
$ !
$ et/nodisplay/command=sys$input                ! Substitute the activity you want
QUIT
$ end_cpu=f$getjpi(0,"CPUTIM")                  ! Get ending stats
$ end_dio=f$getjpi(0,"DIRIO")
$ end_pf=f$getjpi(0,"PAGEFLTS")
$ run sys$manager:hard                          ! Get Hard page faults
$ end_hard = f$integer(hardflts)                ! HARD sets HARDFLTS symbol
$ min=f$cvtime(,,"MINUTE")
$ sec=f$cvtime(,,"SECOND")
$ hun=f$cvtime(,,"HUNDREDTH")
$ end_et=hun + (sec*100) + (min*60*100)
$ usd_cpu = end_cpu - start_cpu                 ! Calculate used stats
$ usd_dio = end_dio - start_dio
$ usd_hard = end_hard - start_hard
$ usd_pf = (end_pf - start_pf) - usd_hard       ! Soft faults = total-hard
$ usd_et = end_et - start_et
$ out = f$fao ("Cpu: !5UL Dio: !5UL SF: !5UL HF:!5UL ET:!5UL",usd_cpu,-
  usd_dio,usd_pf, usd_hard,usd_et)
$ write sys$output "''out'"                     ! Output stats
$ goto loop
```

Unfortunately, there is no way to extract hard page fault statistics using F$GETJPI; thus BENCH executes a small macro program (HARD.MAR) to retrieve this information, as follows:

```
$ TYPE HARD.MAR
;
; HARD.MAR - Get processes hard page fault count and store in HARDFLTS symbol
;
; To compile, issue the following commands:
;    $ MAC HARD
;    $ LINK HARD
;
;    Put in same directory as BENCH.COM
;
     .EXTRN LIB$SET_SYMBOL
     .MACRO CHECK_STATUS ?OK
     BLBS R0,OK
     $EXIT_S   R0
OK:  .ENDM CHECK_STATUS

     .LIBRARY        /SYS$LIBRARY:LIB.MLB/
     .LINK           /SYS$SYSTEM:SYS.STB/
     $PHDDEF
```

```
;
SYMBOL:         .ASCID "HARDFLTS"               ;Symbol Name
FMT1:           .ASCID          "!10UL"
HARD_BUF        .LONG 20
                .ADDRESS 10$
10$             .BLKB 20

GLOBAL_SYM      .LONG LIB$K_CLI_GLOBAL_SYM ; Select Global symbols tbl
HARD:           .BLKL 1
;
  .ENTRY GET_HARD,^M<R2,R3>

  MOVL G^CTL$GL_PHD,R3
  MOVAL HARD, R2
  MOVL PHD$L_PGFLTIO (R3),(R2)+

  $FAO_S CTRSTR=FMT1,OUTBUF=HARD_BUF,OUTLEN=HARD_BUF,P1=HARD
  CHECK_STATUS
  PUSHAL GLOBAL_SYM ; Make it a GLOBAL symbol
  PUSHAL HARD_BUF         ; Push address of symbol value
  PUSHAL SYMBOL      ; Push address of symbol name
  CALLS #3,G^LIB$SET_SYMBOL    ; Define it
  CHECK STATUS

  MOVL #SS$_NORMAL,R0
  RET
  .END GET_HARD
```

The source code for BENCH.COM and HARD.MAR is contained on the diskette included with this book (see Appendix E).

When the foregoing procedure is executed on a VAXstation 2000, the following is displayed:

```
$ @BENCH/OUTPUT=BENCH.LOG
$ TYPE BENCH.LOG

Cpu: 589        Dio: 11         SF: 928         HF: 170         ET: 5528
Cpu: 522        Dio: 12         SF: 1081        HF: 103         ET: 4649
Cpu: 532        Dio: 12         SF: 1062        HF: 100         ET: 4941
```

The abbreviations in the foregoing output represent the following:

```
CPU - CPU time used in 1/100ths of a second   ! e.g. 528 = 5.28s
DIO - Number of disk I/O operations
SF - Total number of page faults
HF - Total number of hard page faults
ET - Elapsed time in 1/100ths of a second     ! e.g. 4941 = 49.41s
```

As you can see, elapsed time (response time) to invoke the Text Processing utility (TPU) on this system ranges from 49.41 to 55.28 seconds, which does not meet the objective of 10 seconds listed in Table 18.1. The remaining sections in this chapter analyze the results of this example and present techniques for achieving the required time.

18.5 Identifying Resource Dependencies

After you identify a performance problem, you have to decide on which system resource to focus. Ideally, you should focus on the resource for which users spend most of their time waiting. The procedures for *roughly* estimating CPU, memory, and disk dependency from the usage statistics that you've collected are discussed in the following sections.

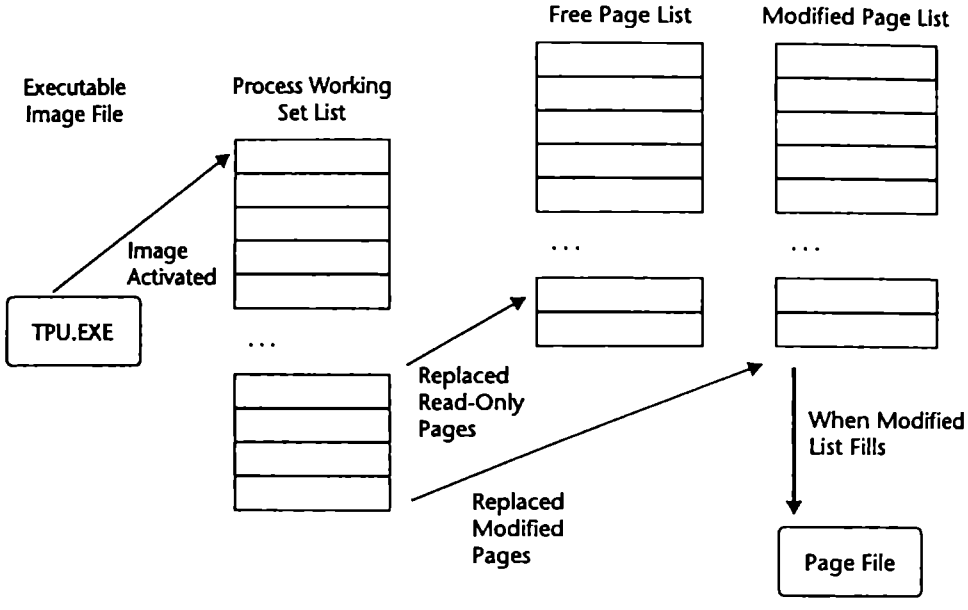
18.5.1 Estimating Memory Dependency

Available memory on a OpenVMS system is organized as a linked list of pages known as the *free page list*. When a process requests memory, pages are removed from the head of the free list and assigned to the process.

Each process has an associated *working set list* that specifies the pages of memory resident in physical memory and that the process can immediately access. The maximum size of the working set list of a process is generally configured with Authorize parameters (e.g., WSQUOTA.WSEXTENT). By restricting working set sizes to a few hundred or a few thousand pages, you can ensure that no one process consumes all the physical memory of a system.

When a process attempts to access more pages of memory than its working set list will allow, *page replacement* occurs. Replacement involves removing older pages from the working set and replacing them with the newly accessed pages. When a page containing read-only structures (e.g., executable code) is replaced, it is placed at the end of the free list. Replaced pages are likely to remain on the free page list for some time before being allocated for another use; as a result, they act as a read-only page cache. When a page containing writable structure (e.g., a data page) is replaced, it is placed on a second list known as the *modified list*. The modified list can grow to contain several hundred or even several thousand pages (depending on the configuration of system parameters) and thus acts as a modified page cache. If the size of the modified list exceeds its configured limits, groups of pages are written to the page file contained on disk.

Figure 18.3 Memory Management Summary



Most of the strain on the memory resource occurs when a process attempts to access a page of memory that is not currently contained it its working set. This event is referred to as a *page fault* and is categorized as either a *hard fault* or *soft fault*, depending on whether disk I/O is required to resolve the fault. Hard faults are resolved either by reading a page from an image file on disk (e.g., XX.EXE) or from a page file. Soft faults are resolved by accessing either the free page or modified page list. Figure 18.3 summarizes the transition of memory pages from one location to another.

When a process activates an executable image, hard page faults occur as an image is loaded from an image file on disk. As the process begins to execute the image, it typically requests memory to store data. This generates soft page faults that are resolved by allocating pages from the free list to the working set of the process. Eventually, the working set of the process reaches its maximum size, forcing read-only or data pages or both to be removed from the working set and placed on the free list and modified list, respectively.

If the process later accesses a replaced read-only page while the page is still on the free list, it is soft faulted back into the working set of the process. If the replaced page is no longer on the free list, the page is hard faulted back into the working set by being read from the original image file. When a process

accesses a replaced data page that is still on the modified list, the page is soft faulted back into the working set of the process. However, if the replaced page has already been removed from the modified list and written to a page file, a hard fault occurs and the page must be read from the page file on disk.

Because soft faults do not require disk I/O, the time to process them depends largely on the CPU. Hard faults depend mostly on the response time of the disk that contains the image file or page file to which the faults occur. You can use the following formula to estimate the amount of elapsed time an activity spends waiting for memory operations:

```
Memory wait time  = Soft fault rate * Soft fault CPU time +
                    Hard-fault-rate * Hard-fault-wait-time
```

The amount of time it takes to process a soft page fault (soft fault CPU time) depends on the speed of the processor. Digital indicates that it takes approximately 0.0005 seconds[1] to process a soft fault on a VAX 11/780 (a 1.0 VUP processor; VUP means VAX units performance, a relative performance measurement). You can extrapolate the time it takes to process as follows:

```
Soft fault CPU time
     = Soft fault rate * 0.0005 sec * (1/Processor VUP rating)
Soft fault CPU time
     = Soft fault rate * 0.0005 sec * (1/12)
     = Soft fault rate * 0.00004 sec              ! On a 12 VUP VAX
```

Because a hard page fault requires disk I/O, the time to process a hard fault is roughly equal to the service time of the device. The procedure for calculating disk service time is discussed in the next section.

18.5.2 Estimating Disk Input-Output Dependency

You can calculate the amount of time an activity spends waiting for disk operations to be completed by using the following formula:[2]

Disk wait time = I/O count * Disk service time

Disk service time varies from one device to another, but you can approximate a value (in milliseconds) using the following formula:

Disk service time (msec) = (Average queue length /
Average operation rate) * 1000

Measure the operation rate and queue length for the disk during a time you believe represents normal usage, as follows:

```
$ MONITOR DISK/ITEM=ALL/BEGIN=0/END="+0-00:03:00"/ NODISPLAY /SUMMARY=DISK.SUM
$ TYPE DISK.SUM

                          VAX/VMS Monitor Utility
                            DISK I/O STATISTICS
                               on node BIRCH      From: 19-SEP-1995 18:05:24
                                  SUMMARY           To: 19-SEP-1995 18:08:26

I/O operation rate                 CUR        AVE        MIN        MAX

BIRCH$DUA0:     VAXVMSV055        4.30       5.42       3.93       7.30
BIRCH$DUA1:     VAXVMST060        0.00       5.87       2.50       9.81

                              SUMMARIZING
                          VAX/VMS Monitor Utility
                            DISK I/O STATISTICS
                               on node BIRCH      From: 19-SEP-1995 18:05:24
                                  SUMMARY           To: 19-SEP-1995 18:08:26

I/O Request Queue Length           CUR        AVE        MIN        MAX

BIRCH$DUA0:     VAXVMSV055        0.00       0.32       0.00       1.32
BIRCH$DUA1:     VAXVMST060        0.00       0.29       0.00       0.98

   SUMMARIZING
```

Then estimate service time by using the average operation rate and queue length. For example, the estimated response times for DUA0 and DUA1 are as follows:

```
DUA0: (.32/5.42) * 1000 = 59 ms
DUA1: (.29/5.87) * 1000 = 49 ms
```

Use the estimated response times for disks containing page files to calculate hard fault wait time.

18.5.3 Estimating CPU Dependency

The following two factors account for elapsed time when processing CPU requests:

- Time waiting for the CPU to become available
- Actual time performing computer operations

When a process is scheduled to receive CPU time, it receives an allocation of CPU time based on the system parameter QUANTUM. On many systems QUANTUM is set to a default value of 20, which represents 0.2 seconds. Once a process begins using its quantum, it is allowed to continue until one of the following events occurs:

- Another process with a higher priority becomes computable
- The process enters a voluntary wait state (e.g., issues an I/O request)
- The process has consumed its entire quantum

Once a process consumes its quantum (reaches quantum-end), another computable process with the same or higher scheduling priority is given a quantum. If there are no computable processes, a single process can continue to receive a new quantum after each quantum-end. As you can see, if there are a large number of computable processes, the amount of time a single process has to wait to use the CPU can be considerable. For example, if quantum is 200 msec and there are 10 computable processes, each process has to wait 2 seconds (10 * 200 msec) each time it needs to use the CPU.

You can calculate average CPU waiting time at a given point in time using the following formula:

```
CPU wait time = (Average computable processes - 1) * Quantum
```

You can measure the average number of computable processes by noting the average Compute queue length using MONITOR, as follows:

```
MONITOR> MONITOR STATES/ALL

                    VAX/VMS Monitor Utility
                        PROCESS STATES
                        on node BIRCH
                    4-JUN-1995 13:38:56

                              CUR       AVE       MIN       MAX

Collided Page Wait           0.00      0.00      0.00      0.00
Mutex & Misc Resource Wait   0.00      0.00      0.00      0.00
Common Event Flag Wait       0.00      0.00      0.00      0.00
Page Fault Wait              0.00      0.05      0.00      1.00
Local Event Flag Wait        0.00      0.11      0.00      1.00
Local Evt Flag (Outswapped)  0.00      0.00      0.00      0.00
Hibernate                    6.00      4.94      3.00      6.00
```

```
Hibernate (Outswapped)          5.00      6.05      5.00      8.00
Suspended                       0.00      0.00      0.00      0.00
Suspended (Outswapped)          0.00      0.00      0.00      0.00
Free Page Wait                  0.00      0.00      0.00      0.00
==>Compute                      2.00      2.00      1.00      2.00
Compute (Outswapped)            0.00      0.00      0.00      0.00
Current Process                 1.00      1.00      1.00      1.00
```

From the foregoing output, the average CPU wait time (per quantum) is as follows:

```
CPU wait time = (2.00 - 1.00) * 0.200 sec = 0.200 sec
```

The total time spent waiting for the CPU during a particular activity is the average wait time multiplied by the total number of quanta needed. This can be roughly calculated as follows:

```
Total CPU wait time
= CPU wait time * total quanta used
= CPU wait time * [(total CPU time/QUANTUM) + hard faults + disk I/Os]
```

Hard faults and disk I/Os are added because they force an immediate quantum-end. Last, you estimate the time the activity spends performing other computer operations as follows:

```
CPU compute time = Total CPU time - Soft fault CPU time
```

18.5.4 Resource Dependency Example

In an earlier section we said that invoking the TPU on a VAXstation 2000 used the following resources:

```
Cpu: 532    Dio: 12    SF: 1062    HF: 10    ET: 4941
```

Using the formulas that have been presented, you can estimate the waiting times for each resource.

```
Total soft fault CPU time
     = Soft-fault-rate * 0.0005 sec * (1/Processor VUP rating)
     = 1062 * 0.0005 * (1/0.75)          ! VS2000 = 0.75 VUPs
     = .708 sec

Total hard fault wait time
     = Hard fault count * Disk service time
     = 100 * 0.049 sec                   ! RD54 disk
     = 4.9 sec
```

Table 18.2 Example Resource Dependency Analysis

System Resource	*Calculated Wait Time (sec)*	*Percentage*
Memory		
Soft faults	0.708	1.6
Hard faults	4.900	11.2
Disk	5.880	13.4
CPU		
Wait time	27.720	63.3
Compute time	4.612	10.5
Total	43.820	100.0

```
Total disk wait time
     = I/O count * Disk service time
     = 12 * .049 sec
     = 5.88 sec

Total CPU wait time
     = CPU wait time * [(Total CPU time / QUANTUM) + Hard faults + Disk I/Os]
     = 0.200 * [(5.32 / 0.200) + 12 + 100]
     = 27.72 sec

CPU compute time
     = Total CPU time - Soft fault CPU time
     = 5.32 sec - 0.708 sec
     = 4.612 sec
```

The results for the TPU example are summarized in Table 18.2.

As you can see, the blocking resource is the CPU, which accounts for nearly 75% of the total elapsed time, 63% of which is spent simply waiting to use the CPU. As a result, the only way to achieve a 10-second response time is to focus on reducing CPU wait time.

You'll notice that the total calculated wait time (43.820 sec) does not exactly match the actual elapsed time recorded by BENCH (49.41 sec). This is because the formulas presented in the previous sections are only rough approximations. However, if you find that the calculated and actual elapsed times vary dramatically, you should verify all calculations.

18.6 Load Balancing

When you have a performance problem and you have identified the blocking resource, the next step is to consider load-balancing options.

Load balancing activities include the following:

- Distributing load among systems in a VMScluster
- Shifting high-priority interactive activities to lower-priority batch jobs
- Delaying processing to off-peak periods (e.g., night or weekends)

Many computer systems sit idle from 5 p.m. to 9 a.m. This idle time represents more than 60% of the total potential capacity of a system. While you certainly can't expect users to wait until midnight to edit an important document, you should carefully identify less critical system activities that can be performed off-hours as batch jobs. This will free scarce resources for users during normal business hours.

If you have multiple stand-alone systems, or a VMScluster, you should ensure that user workloads are distributed as evenly as possible across available systems. You may have one system that is fully utilized and another system that is only 50% utilized. By distributing the load equally you can increase overall responsiveness. If you do not have multiple systems to distribute load or if your systems are already load-balanced, skip to the sections of this chapter on *tuning*.

The following sections discuss techniques for load balancing each of the major system resources.

18.6.1 Load Balancing Memory

To load balance memory you should try to ensure that page faulting and memory use are distributed proportionally among available systems. Monitor memory use on each system (during peak times) using the following command:

```
$ MONITOR FAULT/BEGIN=0/END="+0-00:03:00"/NODISPLAY/SUMMARY=FAULT.SUM

$ TYPE FAULT.SUM
                    VAX/VMS Monitor Utility
                 PAGE MANAGEMENT STATISTICS
                      on node BIRCH      From: 19-SEP-1995 18:37:19
                         SUMMARY           To: 19-SEP-1995 18:40:19
```

```
                              CUR        AVE        MIN        MAX

Page Fault Rate              2.93       5.47       0.00      53.42
Page Read Rate               0.00       0.16       0.00       2.28
Page Read I/O Rate           0.00       0.07       0.00       0.98
Page Write Rate              0.00       0.00       0.00       0.00
Page Write I/O Rate          0.00       0.00       0.00       0.00

Free List Fault Rate         0.00       1.26       0.00       9.12
Modified List Fault Rate     2.93       2.54       0.00      20.52
Demand Zero Fault Rate       0.00       0.53       0.00       7.84
Global Valid Fault Rate      0.00       1.05       0.00      15.25
Wrt In Progress Fault Rate   0.00       0.00       0.00       0.00
System Fault Rate            0.00       0.00       0.00       0.00

Free List Size            2933.00    2871.47    2852.00    2975.00
Modified List Size         220.00     214.86     184.00     251.00

SUMMARIZING
```

If one system has consistently more free memory (free list size) than another, you might shift memory-intensive users or batch jobs to that system. Similarly, if one system incurs considerably more page faults than another, you might want to try to equalize faulting among available systems.

18.6.2 Load Balancing Disk I/O

Most Digital disks can handle 20–60 I/Os per second. If you exceed the capacity of a drive, disk queue lengths and service times increase. If you have an I/O-bound application that is performing poorly and the disk it is accessing has a large queue length, you should consider distributing I/O to less utilized disks.

Because of the large number of files on the system disk, it is frequently overutilized. If you experience this problem, you can migrate OpenVMS data and log files to alternative disks, including the following:

- Queue database
- System authorization file
- Security audit journal
- Resource accounting database

The procedures for locating these files on alternative disks are discussed in detail in chapter 5.

18.6.3 Load Balancing CPU

To gauge CPU activity use Monitor as follows:

```
$ MONITOR MODES/ALL

                    VAX/VMS Monitor Utility
                    TIME IN PROCESSOR MODES
                         on node BIRCH
                      4-JUN-1995 17:07:25

                        CUR       AVE       MIN       MAX

Interrupt Stack        6.36      7.97      3.49     11.46
MP Synchronization     0.00      0.00      0.00      0.00
Kernel Mode           21.01     23.87     21.01     27.61
Executive Mode        20.70     17.79     14.33     20.70
Supervisor Mode       26.75     22.92     17.14     26.75
User Mode             20.06      9.37      2.53     20.06
Compatibility Mode     0.00      0.00      0.00      0.00
Idle Time              5.09     18.04      5.09     31.11
```

Here an idle time of only 8% indicates the CPU is very busy. For batch oriented systems, idle times near zero are acceptable (if not desirable). However, to yield consistent response on time-sharing systems, Digital recommends that you maintain no less than 30% idle time. If you have some systems that are mostly idle and others that are mostly busy, you should move system activity from busy systems to systems that are relatively idle.

To balance the CPU load generated by batch jobs, you can establish execution batch queues on each system and have those queues fed by a single cluster-wide or network-wide generic queue. This process is discussed in detail in chapter 8.

To balance interactive CPU load, you should try to balance logins across multiple systems. This can be difficult if users log in from direct terminal ports (e.g., TXA0). However, if you use terminal servers, you can have OpenVMS perform automatic load-balancing using LAT service ratings and VMScluster alias node names. With this technique, you establish an alias node name for the cluster. The LAT software running on the cluster continuously determines which system is more utilized than the other. When users connect to the alias service, they are attached to the cluster member that is least busy. For more information on establishing a VMScluster alias name, consult the OpenVMS networking manuals.

18.7 Tuning Memory

If you've determined that the memory resource is a performance bottleneck and you have considered load balancing options, the next step is to tune the memory subsystem.

18.7.1 Introduction

Digital systems typically have between 4 and 512 megabytes of physical memory. OpenVMS allocates memory in pages of 512 bytes each. You can translate megabytes to pages by multiplying by 2000. For example, a system with 16 megabytes of memory has approximately 32,000 pages. With the introduction of OpenVMS 6.0, OpenVMS supports up to 3.5 gigabytes (GB) of physical memory on the VAX 6600, 7600, and 10000 systems.

[OpenVMS AXP 1.5]

AXP OpenVMS systems use a different page size than VAX OpenVMS systems. An AXP page size can be either 8K, 16K, 32K, or 64K bytes (all current systems use an 8K page). To maintain compatibility with VAX systems, AXP systems define a unit of page measurement called a *pagelet*, which equals one VAX page. For example, on an AXP system with an 8K page, one AXP page equals 16 pagelets.

On an AXP system, usually all references to a quantity of pages are displayed and evaluated in terms of pagelets (e.g., GBLPAGES, PGFLQUOTA). On the other hand, all references to physical memory are displayed and evaluated in terms of AXP pages (e.g., PHYSICALPAGES, SHOW MEMORY).

The following sections describe how to approach excessive hard and soft page faulting problems.

18.7.2 Excessive Hard Page Faults

A high hard fault rate is strong evidence that either your memory subsystem is severely mistuned or an application is poorly designed. An excessive hard page fault rate could be caused by one or more of the following problems.

Excessive Image Activation

When a process activates an image, hard faults occur as the executable image is read from the image file into memory. Because the execution of many DCL commands results in an image activation, complex DCL command proce-

dures can easily generate hundreds of image activations per minute. If you have such procedures, you should consider streamlining them, or if possible, implement them with high level language.

You can also reduce image activation overhead by installing images shareable with the OpenVMS Install utility. When an executable image is installed as shareable, only the first process to access it incurs hard page faults as it reads the image and sets up global sharing structures. Subsequent processes accessing the image concurrently incur global soft page faults because they can access the same pages already mapped into memory by the previous process. To determine which images to install as shareable you can use image-level accounting to identify frequently activated images, as follows:

```
$ ACCOUNTING/TYPE=IMAGE/SUMMARY=IMAGE/OUTPUT= -
                IMAGE.LIS/REPORT=(RECOPRDS,PAGE_READS) ACCOUNTING

$ TYPE IMAGE.LIS

From: 1-OCT-1995 21:04          VAX/VMS Accounting Report
To: 1-OCT-1995 22:21

Image name      Total      Page IO
                Records    Reads
--------------------------------
ACC             1          14
CREATE          3          21
DELETE          2          13
DIRECTORY       1          11
EDT             5          31
LINK            4          39
LOGINOUT        14         109
MACRO32         5          144
MONITOR         5          138
QUEMAN          3          35
SET             5          40
SETPO           2          6
SHOW            12         66
SUBMIT          7          71
SYSGEN          2          45
TPU             35         212
TYPE            9          61
VMSHELP         1          42
```

The images with the most activations (total records) and the most hard faults (page IO reads) are the best shareable image candidates. In this case, TPU is one possible candidate.

Bear in mind that an executable image may also access shareable libraries that could benefit from being installed. You can identify the shareable libraries associated with an image by performing the following command:

```
$ ANALYZE/IMAGE/OUT=T.TMP TPU.EXE
$ SEARCH T.TMP/WINDOW=(0,10) "Shareable Image List"

Shareable Image List
0) this image
1) "LIBRTL"
2) "MTHRTL"
3) "VAXCRTL"
4) "TPUSHR"
```

You may also want to use the undocumented SET WATCH command to see other types of files that are accessed when you invoke an image, as follows:

```
$ SET WATCH FILE/CLASS=ALL
$ EDIT/TPU
...
%XQP, Directory scan for: TPU$CCTSHR.EXE;0, Status:00000000
...
%XQP, Directory scan for: EVE$SECTION.TPU$SECTION;0, Status:00000000
...
$ SET WATCH FILE/CLASS=NONE
```

As you can see, SET WATCH uncovers two additional files that are accessed when TPU is invoked. These files are thus potential Install candidates.

To install an image or library as shareable use the Install utility as follows:

```
$ INSTALL:==$INSTALL/COMMAND_MODE
$ INSTALL ADD SYS$SYSTEM:TPU.EXE/OPEN/SHARE
$ INSTALL ADD SYS$LIBRARY:TPUSHR.EXE/OPEN/SHARE
```

Some of the foregoing libraries may already be installed by OpenVMS. To make these changes permanent, you should add the foregoing commands to your system startup procedures, as discussed in chapter 5 (e.g., STARTUP_SHARE.COM).

To install a shareable image you need sufficient global pages and global sections. To determine the number available, perform the following steps:

```
$ INSTALL:==$INSTALL/COMMAND_MODE
$ INSTALL LIST/GLOBAL/SUMMARY
```

```
                Summary of Local Memory Global Sections

186 Global Sections Used, 13032/16968 Global Pages Used/Unused
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> PARAMETER SHOW GBLSECTIONS

%SYSMAN-I-USEACTNOD, a USE ACTIVE has been defaulted on node BIRCH
Node BIRCH: Parameters in use: ACTIVE

Parameter Name Current   Default   Minimum   Maximum   Unit Dynamic
-------------------------------------------------------------------
GBLSECTIONS    400       250       60        4095      Sections
```

The number of unused global pages is displayed directly—in this example, 16,968. To determine the number of available global sections, you must subtract the number in use from the system parameter GBLSECTIONS. In the foregoing example there are 214 free global sections (400 - 186). If you do not have many free global pages or global sections, you should add more using AUTOGEN (discussed later in this chapter).

Image File I/O

If you have applications that don't warrant being installed as shareable (e.g., there is little concurrent access), users may generate disk I/O to the image file during or even after image activation. This can be caused by one or more of the following problems:

- Insufficient free list cache size
- Insufficient working set size

If users' working sets are too small, code pages are frequently replaced and are placed on the free list. If the free list cache is not large enough, these cached pages are quickly allocated to other processes before they are faulted back into users' working sets. If a MONITOR PAGE indicates a small free list size and a low free list fault rate, you should consider adjusting the following system parameters:

FREELIM: Specifies the minimum allowable size of the free list. When the number of pages on the free list falls below FREELIM, OpenVMS attempts to reclaim memory from processes until FREEGOAL pages are on the free list.

FREEGOAL: Specifies the target free list size.

Because FREELIM specifies the minimum size of the free list, it also represents the minimum number of read-only pages cached. You should increase FREELIM to a value that provides efficient caching of read-only pages. As an alternative, if only a few specific processes are experiencing high hard fault rates, you might achieve better results by increasing their individual working sets (see the following sections).

Page File I/O

Another reason for high hard page fault rates is faulting modified pages from a page file. The MONITOR PAGE command allows you to monitor all hard fault activity based on the Page Read I/O statistic. Unfortunately, there is usually no easy way to determine exactly what portion of the I/Os are reading from files versus reading from page files. However, if your page files are located on dedicated disks, you can gauge page file activity by reviewing the results of MONITOR DISK statistics.

You can reduce hard faults resulting from page file reads by increasing the maximum allowable size of the modified page list. The modified page list size and other attributes are configured with the system parameters shown in Table 18.3.

CAUTION: Before you increase the size of the modified list, be certain that there are no processes whose working sets are configured so low that they generate the majority of modified pages.

Establishing Page File Sizes

When pages are written from the modified list, OpenVMS must locate a contiguous area within the page file. If a contiguous area cannot be located, OpenVMS attempts to write pages in several smaller operations. When the page file becomes full, OpenVMS incurs considerable overhead trying to locate page file space, during which time processes trying to add pages to the modified list may be placed into a wait state. If the situation becomes severe enough, the entire system can hang. As a result, Digital recommends that you maintain page files with at least 50% free space to minimize page file fragmentation.

You can monitor page file allocation with the following command:

```
$ SHOW MEMORY/FILE
```

Table 18.3 Modified Page Writer (MPW) System Parameters

Parameter	*Description*
MPW_WAITLIMIT	Forces processes into a wait state if they attempt to add pages to the modified list when it contains more than MPW_WAITLIMIT pages. Such processes are not allowed to continue until the modified page writer completes writing pages.
MPW_HILIMIT	Specifies the number of pages that can accumulate on the modified list before OpenVMS begins writing modified pages to the page file. After pages are written to the page file, they are returned to the end of the free list.
MPW_THRESH	Minimum pages that must be contained on the modified list for OpenVMS to consider writing modified pages to the page file.
MPW_LOWAITLIMIT	Processes that are forced into a wait state because of modified page writing are allowed to resume when the number of pages on the modified list drops below MPW_LOWAITLIMIT.
MPW_LOLIMIT	When OpenVMS determines that it is necessary to write pages from the modified list to the page file, it continues until only MPW_LOLIMIT pages remain. In this way, MPW_LOWLIMIT specifies the minimum size for the modified list cache.

```
                System Memory Resources on 5-OCT-1995 11:45:38.83

Paging File Usage (pages):                      Free      Reservable      Total

DISK$VAXVMSV055:[SYS0.SYSEXE]SWAPFILE.SYS       1736        1736          2096
DISK$VAXVMSV055:[SYS0.SYSEXE]PAGEFILE.SYS       3610       -18224         5600
```

If you see that free space is less than 50% of total space, you should increase the page file size using SYSGEN as follows:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> CREATE SYS$SPECIFIC:[SYSEXE]PAGEFILE.SYS/SIZE=6000
%SYSGEN-I-EXTENDED, SYS$SPECIFIC:[SYSEXE]PAGEFILE.SYS;1 extended
```

You should also note the number of *reservable* pages indicated by the SHOW MEMORY command. If your system has multiple page files, OpenVMS uses the reservable page count to determine page file assignment. When a process is created, OpenVMS examines the reservable page count of

each page file. The process is assigned an allocation of up to 2048 pages (the default value of the system parameter RSRVPAGCNT) in the page file that has the most reservable pages. If a process uses up its allocation, OpenVMS examines the reservable pages a second time and gives the process another allocation in the page file that has the most available pages. As a process continues to use its allocations, it can be assigned to up to four different page files. Thus, if you experience extremely heavy page file activity you can increase performance by creating several page files on different disks.

On many systems the number of reservable pages is frequently negative. This occurs when many small processes are given an allocation of 2048 pages but only use a small portion of it. In the foregoing example, the reservable page count was about -18,000 because there were approximately 12 active processes.

18.7.3 Excessive Soft Page Faults

Excessive soft faults are less of a concern than hard faults because they are resolved 100 times faster. However, if your applications are generating many hundreds of soft faults per second, you may be able to increase performance by tuning working set related system parameters.

You must consider the following three mechanisms when tuning working sets:

- Working set sizes
- Automatic working set adjustment (AWSA)
- Memory reclamation

18.7.3.1 Establishing Working Set Sizes

The *working set list* of a process is an array of pointers that describes the physical pages of memory that a process can access without incurring a page fault. The purpose of the working set mechanism is to limit the amount of physical memory any one process can control. Working set list size is controlled by the following three parameters:

- Working set default (also called working set limit)
- Working set quota
- Working set extent

When a process activates an image, its working set list size is set to the *default* value. As the process generates page faults and maps physical pages, its working set list size can increase to the *quota* value. The working set list of the process can expand further, up to the *extent* value, but only if sufficient free memory is available.

When a process needs to reference more pages than can be mapped by its working set, pages must be removed from the working set. The removed pages migrate to the free list (as described earlier). If migrated pages are referenced again, they are faulted back into the working set, forcing the replacement of other pages. When the actual working set size of a process is configured far below what it actually requires, pages are faulted back and forth between the working set and the system page caches. If you detect an application that faults heavily, you should try to expand its working set until page fault rate levels off.

The mechanisms used to configure working sets depend on process type, as follows.

Interactive Processes

Working set limits are established from the UAF, as follows:

```
/WSDEFAULT      Working Set Default       ! Default 150 pages
/WSQUOTA        Working Set Quota         ! Default 256 pages
/WSEXTENT       Working Set Extent        ! Default 512 pages
```

The limit will not be set lower than the following system parameters:

```
PQL_MWSDEFAULT Minumum Working Set Default    ! Default 60 pages
PQL_MWSQUOTA   Minimum Working Set Quota      ! Default 60 pages
PQL_MWSEXTENT  Minimim Working Set Extent     ! Default 60 pages
```

Detached Processes

Working set limits are established from the following system parameters:

```
PQL_DWSDEFAULT Default Working Set Default    ! Default 60 pages
PQL_DWSQUOTA   Default Working Set Quota      ! Default 60 pages
PQL_DWSEXTENT  Default Working Set Extent     ! Default 60 pages
```

The limit will not be set lower than corresponding PQL_M* system parameters.

Table 18.4 Working Set Parameters

Working Set	*Small Jobs*	*Medium Jobs*	*Large Jobs*
Default	512	512	1024
Quota	768	1024	2048
Extent	1024	2048	4096

Batch Processes

Batch queues may be configured with working set characteristics using the INITIALIZE/QUEUE,START/QUEUE commands and the following qualifiers:

```
/WSDEFAULT=pages     Working Set Default
/WSQUOTA=pages       Working Set Quota
/WSEXTENT=pages      Working Set Extent
```

If these queue characteristics are not specified, then working set parameters are set to the working set limits established in the UAF, subject to the limits of the PQL_D and PQL_M system parameters.

In general, the standard working set parameters are set much too small for the average application. Table 18.4 contains some guidelines for establishing working set parameters.

If a system has ample free memory, you can increase working sets substantially higher than indicated in Table 18.4. Ideally, you should distribute available memory to applications that will most benefit from it. Generally, these are applications that exhibit the highest page fault rates.

18.7.3.2 Automatic Working Set Adjustment

After a process invokes an image, working set size adjustments are controlled by the AWSA mechanism. AWSA behavior is determined by the configuration of the system parameters listed in Table 18.5.

By default, every 200 msec (AWSTIME), AWSA is triggered and OpenVMS calculates a page fault rate for a process. If the fault rate is greater than the system parameter PFRATH, the working set size of the process is increased by WSINC entries. The working set size of a process can increase up to its working set quota value without regard to free memory. However, working set size can increase beyond its quota (up to its extent) only if there are at least

Table 18.5 Working Set Adjustment

Parameter	*Description*
AWSTIME	(Default 20 = 200 msec) The interval at which working set adjustment occurs
PFRATH	(Default 120 = 12 faults/sec) Minimum page fault rate necessary to justify a working set size increase
WSINC	(Default 150 entries) Amount by which working set size is increased when PFRATH is triggered
PFRATL	(Default 0 = Disabled) Forces working set size decrease if page fault rate drops below specified value
WSDEC	(Default 250 entries) Amount by which working set size is decreased when PFRATL is triggered
BORROWLIM	The minimum number of pages that must exist on the free list for working set list size of a process to be expanded above its quota value
GROWLIM	The minimum number of pages that must exist on the free list for a process to map additional pages of memory into a working set larger than the working set quota of the process

BORROWLIM pages on the free list. In addition, a process can map more pages than its working set quota only if there are at least GROWLIM pages on the free list.

You can enable AWSA as an optional feature to automatically decrease working set list sizes by setting the system parameters PFRATL and WSDEC to non-zero values. When AWSA is enabled, the working set size of a process is decremented if its page fault rate drops below PFRATL. However, automatic working set decrementing generally causes undesirable working set size oscillation and is therefore not normally enabled.

18.7.4 Memory Reclamation

Up to this point, we have discussed how OpenVMS allocates memory to processes by increasing their working set sizes. Once memory becomes scarce, OpenVMS uses the following mechanisms to reclaim memory from idle processes and processes that have exceeded their working set quotas:

- Swapper Trimming
- Proactive memory reclamation (OpenVMS 5.4-3)

18.7.4.1 Swapper Trimming

When the number of pages on the free list drops below the system parameter FREELIM, the SWAPPER process attempts to reclaim memory by initiating *swapper trimming*. The swapper calculates the number of pages that must be recovered to establish FREEGOAL pages on the free list. It then begins first-level trimming by reducing process working sets to their quota values until enough pages are reclaimed.

If first-level trimming does not free enough pages, the swapper initiates second-level trimming and trims selected process working sets back to the value of the system parameter SWPOUTPGCNT until sufficient pages are reclaimed. Finally, if trimming efforts do not yield enough free memory, the swapper swaps selected processes to the swap file, making all the remaining pages available in the process. The swapper first trims processes that are least likely to become computable, thereby freeing memory for more active processes.

Swapper trimming is generally desirable because it allows you to maximize use of the system memory. However, if you detect excessive trimming or swapping of active processes, you probably do not have enough physical memory to support existing working set sizes, and thus you should consider decreasing these working sets.

18.7.4.2 Proactive Memory Reclamation

[OpenVMS 5.4-3]

Beginning with OpenVMS 5.4-3, Digital added the proactive memory reclamation (PMR) mechanism. This mechanism is designed to reclaim memory more aggressively—*before* a memory deficit is detected. PMR targets reclamation from the following two types of processes whenever the number of pages on the free list drops below FREEGOAL.

- Long-waiting processes
- Periodically waking processes

Processes that have been in wait states (hibernate and local event flag wait) for longer than LONGWAIT seconds are swapped. Unlike swapper trimming, the working set *sizes* of outswapped processes are not reduced. Therefore, when the process is swapped back in, it is able to begin executing more quickly without excessive page faulting.

Swapper trimming is ineffective in reclaiming memory from periodically waking processes because such processes still consume some amount of CPU

time and are never considered idle. PMR monitors for periodically waking processes whenever the number of pages on the free list drops below two times FREEGOAL. It locates processes that have been idle for more than 30 seconds and that are idle 99% of the time. When such processes are identified, 25% of their pages are trimmed from their working sets; however, process working set sizes are not reduced.

18.8 Tuning I/O

If you determine that disk I/O is constraining an application or the system as a whole, use the strategies discussed in the following sections to improve I/O performance.

18.8.1 Disk Defragmentation

When a disk is first initialized, its available free space is contained in one large contiguous area. When files are created, they are allocated from this contiguous pool and thus are also contiguous. Eventually as more files are created, deleted, and extended, available free space becomes more fragmented. Eventually files must be allocated to space made up of multiple fragments located in different physical regions of a disk. When users perform an I/O operation to a fragmented file, OpenVMS may be forced to break a single operation into two or more operations—*a split I/O*—to read the multiple fragments.

A file is made up of one or more extents. Each *extent* describes each fragment of a file. A contiguous file has only a single extent, whereas a severely fragmented file might have several hundred extents. OpenVMS can access only a limited number of extents within a file at one time. By default, this limit is defined by the system parameter ACP_WINDOW, which has a default value of 7. If a file has more than seven fragments, and a user attempts to access a fragment not described by the current window, OpenVMS must perform a *window turn* and retrieve information about the unmapped fragment. An excessive window turn rate is an indication of a severe fragmentation problem.

You can monitor the system-wide split I/O rate by issuing the following command and reviewing split transfer rate:

```
$ MONITOR IO/BEGIN=0/END="+0-00:03:00"/NODISPLAY/SUMMARY=IO.SUM
$ TYPE IO.SUM
```

```
                    VAX/VMS Monitor Utility
                     I/O SYSTEM STATISTICS
                        on node BIRCH      From: 3-OCT-1995 12:05:52
                          SUMMARY            To: 3-OCT-1995 12:08:53

                          CUR       AVE       MIN       MAX

Direct I/O Rate          1.98      2.05      0.33      3.65
Buffered I/O Rate        0.99      3.33      0.33      6.64
Mailbox Write Rate       0.00      0.00      0.00      0.00
Split Transfer Rate      0.00      0.00      0.00      0.33
Log Name Translation Rate 1.98    14.94      0.66     35.09
File Open Rate           0.33      0.53      0.00      0.99
Page Fault Rate        214.56    318.44     80.79    572.18
Page Read Rate           0.00      3.67      0.00     12.95
Page Read I/O Rate       0.00      1.59      0.00      3.98
Page Write Rate          0.00      1.99      0.00     39.86
Page Write I/O Rate      0.00      0.01      0.00      0.33
Inswap Rate              0.00      0.00      0.00      0.33
Free List Size        2537.00   2935.03   2070.00   3854.00
Modified List Size     403.00    312.53    191.00    435.00

                          SUMMARIZING
```

Monitor the system-wide window turn rate by using the following command and reviewing window turn rate:

```
$ MONITOR FCP/BEGIN=0/END="+0-00:03:0"/NODISPLAY/SUMMARY=FCP.SUM
$ TYPE FCP.SUM

                    VAS/VMS Monitor Utility
                   FILE PRIMITIVE STATISTICS
                        on node BIRCH      From: 3-OCT-1995 12:09:14
                          SUMMARY            To: 3-OCT-1995 12:12:14

                          CUR       AVE       MIN       MAX

FCP Call Rate            2.29      1.87      0.32      2.95
Allocation Rate          0.00      0.18      0.00      0.65
Create Rate              0.00      0.64      0.00      1.63
Disk Read Rate           0.00      0.01      0.00      0.98
Disk Write Rate          0.00      1.26      0.00      2.95
Volume Lock Wait Rate    0.00      0.00      0.00      0.00

CPU Tick Rate            1.31      1.89      0.32      3.93
File Sys Page Fault Rate 0.00      0.28      0.00      0.98
Window Turn Rate         0.32      0.18      0.00      0.65
```

```
File Lookup Rate                1.63        0.73        0.00        1.63
File Open Rate                  0.65        0.53        0.00        0.98
Erase Rate                      0.00        0.04        0.00        0.32

                                SUMMARIZING
```

Unfortunately, you'll need performance monitoring tools (e.g., DECps) to monitor the split I/O and window turn rates of individual disks.

If your applications depend heavily on access to a specific file, you can determine the number of fragments contained in the file using the Dump utility as follows:

```
$ DUMP/HEADER/BLOCK=COUNT:0 CHECK_PROCESSOR.IDX/OUT=T.TMP
$ SEARCH T.TMP "Count"
          Count:     39          LBN:       24027
```

Because only one extent is listed (containing 39 blocks) this file is contiguous. If there is sufficient contiguous space on a disk, you can defragment a file using the COPY command as follows:

```
$ COPY/CONTIGUOUS filename.ext newfile.ext
```

However, to defragment an entire disk you must perform an image backup and restore the disk or use a disk defragmentation utility (available from Digital and third-party vendors).

18.8.2 Adjust Volume Characteristics

By default, when you delete a file, its disk blocks are deallocated but not erased. Sophisticated users can create a new file that allocates previously deallocated blocks and access another user's deleted data. The volume attributes /HIGHWATER and /ERASE_ON_DELETE prevent this access but result in additional overhead to create and delete files. If potential security disadvantages are acceptable, you can increase disk performance by disabling these characteristics on all disk volumes, as follows:

```
$ SET VOLUME/NOHIGHWATER device
$ SET VOLUME/NOERASE_ON_DELETE device
```

To make the change effective you must dismount and mount the volume. To make the change effective on a system disk you must reboot the system. The security implications of disabling these parameters are discussed in more detail in chapter 17.

18.8.3 Adjust ACP (Ancillary Control Process) Caches

OpenVMS maintains caches of file, directory, and extent structures. If these caches are not large enough, unnecessary I/O is incurred. You can determine the effectiveness of the caches by monitoring cache hit rates using Monitor, as follows:

```
$ MONITOR FILE/BEGIN=0/END="+0-00:03:00"/NODISPLAY/SUMMARY=FILE.SUM
$ TYPE FILE.SUM

                        VAX/VMS Monitor Utility
                   FILE SYSTEM CACHING STATISTICS
                          on node BIRCH      From: 3-OCT-1995 12:20:02
                             SUMMARY           To: 3-OCT-1995 12:23:02

                                    CUR        AVE        MIN        MAX

Dir FCB    (Hit %)               100.00     100.00       0.00     100.00
(Attempt Rate)                     0.32       0.86       0.00       1.94
Dir Date   (Hit %)               100.00     100.00       0.00     100.00
(Attempt Rate)                     0.32       0.86       0.00       1.94
File Hdr   (Hit %)               100.00     100.00     100.00     100.00
(Attempt Rate)                     1.31       1.43       0.32       2.95
File ID    (Hit %)               100.00     100.00       0.00     100.00
(Attempt Rate)                     0.32       0.52       0.00       0.98
Extent     (Hit %)               100.00     100.00       0.00     100.00
(Attempt Rate)                     0.32       0.31       0.00       0.98
Quota      (Hit %)                 0.00       0.00       0.00       0.00
(Attempt Rate)                     0.00       0.00       0.00       0.00
Bitmap     (Hit %)                 0.00       0.00       0.00       0.00
(Attempt Rate)                     0.00       0.00       0.00       0.00

                              SUMMARIZING
```

Digital recommends that you attempt to maintain at least a 75% hit ratio to all caches that have a non-zero attempt rate. If you need to increase the size of cache, adjust the following system parameters:

- ACP_SYSACC — Dir FCB
- ACP_DINDXCACHE — Dir FCB
- ACP_DIRCACHE — Dir Data
- ACP_HDRCACHE — File Hdr
- ACP_FIDCACHE — File ID
- ACP_EXTCACHE — Extent

- ACP_EXTLIMIT Extent
- ACP_QUOCACHE Quota
- ACP_MAPCACHE Bitmap

You can also use the AUTOGEN facility to automatically adjust cache sizes based on feedback information (see the AUTOGEN section at the end of the chapter).

18.8.4 Virtual I/O Cache

[OpenVMS 6.0]

Beginning with OpenVMS 6.0, you can use the virtual I/O cache to improve disk performance. The virtual I/O cache is a data cache that stores the most recently accessed disk blocks in memory. When an application accesses the same disk block more than once, subsequent requests are satisfied from the cache, rather than from the physical disk.

The cache is enabled by setting a system parameter as follows:

```
VBN_CACHE_S = 1      ! Enabled
VBN_CACHE_S = 0      ! Disabled
```

The virtual I/O cache can also be used within a VMScluster, with the following restrictions:

- To enable caching you must enable it on every cluster member
- Caching is disabled for any file that is opened for write access by two or more members

To monitor the status of the virtual I/O cache, use the following command:

```
$ SHOW MEMORY/CACHE/FULL

                System Memory Resources on 2-JUN-1995 17:02:08.51

Virtual I/O Cache
        Total Size (pages)      544     Read IO Count               4622
        Free Pages              0       Read Hit Count              1814
        Pages in Use            544     Read Hit Rate               39%
        Maximum Size (SPTEs)    715     Write IO Count              85
        Files Retained          36      IO Bypassing the Cache      0
```

The foregoing statistics indicate the current size of the cache, the number of free pages, and maximum possible size of the cache. Another important statistic is the read hit rate. According to Digital, a 50% hit rate is the point at which the performance gains from caching exceed the overhead required to maintain the cache.

18.9 Tuning CPU

If you've determined that the CPU is the major resource hampering application or system performance, and you've verified that the CPU overhead incurred by the I/O and memory subsystems has been minimized, you should review the following:

- Process scheduling
- Priority boosting
- Quantum
- Decompress system libraries

18.9.1 Process Scheduling

Within an OpenVMS environment, processes share access to the CPU based on the OpenVMS process scheduling algorithm. The most important aspect of this algorithm is the *scheduling priority* of a process. Scheduling priority for interactive users is initially established based on the priority specified in the users' Authorize record. Batch job priority is set to the default base priority established with the INITIALIZE/QUEUE/BASE_PRIORITY command. In general, various processes are assigned the following priorities:

- 1–3 Batch jobs
- 4–7 Interactive users (usually 4)
- 8–6 VMS system processes (swapper, etc)
- 16–32 Real-time processes

OpenVMS always gives precedence to the computable process with the highest priority. Processes with the same priority are serviced in a round-robin manner. It is important to understand that a totally computable priority 5 process completely blocks the CPU from all priority 4 processes. As a result, all interactive users should be set to the same priority (generally prior-

ity 4). Batch jobs can be assigned priorities between 1 and 3, depending on which are more time critical.

If you suspect that the CPU is not being shared equitably among interactive users, use the Monitor utility to identify processes that are capitalizing the CPU:

```
$ MONITOR PROCESS/TOPCPU

                    VAX/VMS Monitor Utility
                    TOP CPU TIME PROCESSES
                        on node BIRCH
                     4-JUN-1995 17:10:29

                       0         25         50         75        100
                       + - - - - + - - - - + - - - - + - - - - + - - - - +
00000044 INF_LOOP      69  ****************************
                       |         |          |          |         |
0000004F BENCH         24  **********
                       |         |          |          |         |
00000030 SYSTEM         2
                       |         |          |          |         |
                       |         |          |          |         |
                       |         |          |          |         |
                       |         |          |          |         |
                       |         |          |          |         |
                       + - - - - + - - - - + - - - - + - - - - + - - - - +
```

Then use the following command to monitor individual processes:

```
$ SHOW PROCESS/CONT/ID=process-id
```

By default, users cannot increase their priority above the /PRIORITY value (*default priority*) established in the UAF (see chapter 9), unless they have the ALTPRI privilege.

18.9.2 Priority Boosting

Each process actually has two priorities, *base priority* and *current priority*. The current priority is used to determine scheduling precedence. By default, the base and current priorities of a process are equal to its default priority. However, OpenVMS temporarily increases the current priority of a process after certain events. The two most important events are disk I/O completion and terminal I/O completion, after which processes receive a priority boost of 2 and 4, respectively.

Priority boosting helps make processes more likely to receive CPU time immediately after I/O completion. Unfortunately, this can cause I/O-bound jobs to capitalize the CPU. Therefore, if you have I/O intensive batch jobs and you don't want them to interfere with interactive users, you should set their base priority to 1 to disable the effects of priority boosting.

18.9.3 Quantum

When a process is selected to receive CPU time, it is allowed to execute for a set number of seconds, defined by the system parameter QUANTUM. By default, QUANTUM is set to 20, representing 200 msec (or 1/5 of a second).

Once a process is executing, it is allowed to continue until one of the following events occurs:

- Another process with a higher priority becomes computable
- The process enters a voluntary wait state (e.g., issues an I/O request)
- The process has consumed its entire quantum

The default quantum value is designed to yield acceptable response time in environments with a mix of interactive and batch processing. If your environment mostly consists of interactive users, you might consider using a smaller QUANTUM value. If you are attempting to support a large number of concurrent interactive users, you may even want to set QUANTUM to its minimum value of 2 (representing 20 msec).

If you run many batch jobs, you may be able to reduce process scheduling overhead by slightly increasing QUANTUM. If your system is used purely for batch processing, you should consider increasing QUANTUM considerably, perhaps by a factor of 2 or 3.

18.9.4 Decompress System Libraries

When you install or upgrade OpenVMS, a number of system libraries are installed in compressed form. Compressed libraries reduce disk space usage by several thousand blocks. However, if users frequently access OpenVMS Help or use compilers that link to compressed shareable libraries, additional CPU overhead occurs, decompressing the libraries as they are accessed. If you believe these libraries are accessed regularly, you should permanently decompress them as follows:

```
$ @SYS$UPDATE:LIBDECOMP
```

18.10 Expanding Capacity

If your load balancing and tuning efforts do not help increase performance to an acceptable level, your only remaining alternative is to expand system capacity. However, before you purchase additional hardware, you should carefully review the information you've collected concerning individual application and overall system resource demand. With this information you should be able to determine which resource most needs expansion. If you expand a resource that is not constraining application performance, the affect will be minimal, and you will waste considerable money.

18.11 Sample Performance Problem

Let's return to the example of invoking TPU. This section uses the techniques discussed in this chapter to rectify the unacceptable response time measured for this particular activity. The results from BENCH have indicated a 49-second response time to invoke TPU:

```
Cpu: 532     Dio: 12      SF: 1062     HF: 100      ET: 4941
```

The resource dependency analysis has indicated that this activity depends mostly on the CPU, most of the time being spent simply waiting because another processes was using the CPU. Consequently, a MONITOR PROCESS/TOPCPU shows the following:

```
$ MONITOR PROCESS/TOPCPU

                    VAX/VMS Monitor Utility
                    TOP CPU TIME PROCESSES
                         on node BIRCH
                      4-JUN-1995 17:10:29

                      0         25        50        75        100
                      + - - - - + - - - - + - - - - + - - - - +
00000044 INF_LOOP     69  ****************************
                      |         |         |         |         |
0000004F BENCH        24  **********
                      |         |         |         |         |
00000030 SYSTEM        2
                      |         |         |         |         |
                      |         |         |         |         |
                      |         |         |         |         |
                      |         |         |         |         |
                      |         |         |         |         |
                      + - - - - + - - - - + - - - - + - - - - +
```

A process INF_LOOP is consuming 69% of the CPU, forcing the TPU activity to wait for the CPU. On further investigation you determine that the INF_LOOP process is a broken DCL program that is infinitely looping. After you stop this process, BENCH results are as follows:

```
Cpu: 597     Dio: 12      SF: 1031     HF: 201     ET: 1723
```

The 17-second response time is much better but still does not meet the objective. A new resource dependency analysis indicates that hard fault wait time represents a large portion of the current response time. To reduce hard faults, TPU and the images it accesses are installed with the /SHARE option. BENCH results are then as follows:

```
Cpu: 509     Dio: 12      SF: 2385     HF: 9       ET: 631
```

Response time is now 6 seconds—4 seconds less than the objective!

It is interesting to note that in some cases installing an image with the /SHARE option can reduce hard faults—even when the image is not accessed concurrently. In the example, hard faults were reduced even though only a single process was invoking TPU over and over again. When a process exits an image that is not installed, the code pages return to the free list. When the process accesses the image again, it still must hard fault from the image file—even if the image pages are on the free list from an earlier activation. When a process exits an image that is installed, the code pages return to the free list, but global page structures that describe the installed image remain intact. If the process activates the same image again, it soft faults the image from the free list.

18.12 Managing System Parameters

When you change system parameters, it is important that you modify them using the correct procedures. The following sections describe how to properly use the two utilities designed for this process (SYSMAN and AUTOGEN).

18.12.1 SYSMAN PARAMETER command

You can manipulate system parameters with either the SYSGEN utility or the SYSMAN PARAMETER command. You should use SYSMAN, however, because it provides additional features and will eventually supersede SYSGEN.

To review system parameters use SYSMAN as follows:

```
$ RUN SYS$SYSTEM:SYSMAN

SYSMAN> PARAMETER SHOW parameter
SYSMAN> PARAMETER SHOW FREELIM

Node BIRCH: Parameters in use: ACTIVE

Parameter Name  Current  Default   Minimum   Maximum   Unit   Dynamic
--------------------------------------------------------------------
FREELIM         40       32        16        -1        Pages
```

By default, the SHOW command displays the *active* settings for the specified parameter. When the system boots, active parameters are loaded from the permanent system parameter database, referred to as *current* parameters. These parameters are stored in the following file:

```
SYS$SYSTEM:VAXVMSSYS.PAR
```

[OpenVMS AXP]

On AXP OpenVMS systems, system parameters are stored in the following file

```
SYS$SYSTEM:ALPHAVMSSYS.PAR
```

You can view *current* parameter values by issuing a PARAMETER USE command, as follows:

```
SYSMAN> PARAMETER USE CURRENT
SYSMAN> PARAMETER SHOW parameter
```

Dynamic parameters can be changed after a system boot and are identified by a D in the Dynamic column. On the other hand, nondynamic parameters can only be made effective by rebooting the system.

Each parameter has an associated *default*, *minimum*, and *maximum* value. Parameters cannot be set to values outside their associated minimums and maximums. The Digital established default values allow virtually any system configuration to boot. When you have a system parameter configuration problem that prevents your system from booting, you can temporarily force the system to use the default parameters by performing a *conversational boot*, as discussed in chapter 4.

Table 18.6 MODPARAMS.DAT Entry Formats

Format	*Description*
parameter = value	Sets the specified parameter to an absolute value
ADD_parameter = delta-value	Sets the parameter equal to the sum of the default value of the parameter plus any additional amount that AUTOGEN calculates is necessary
MIN_parameter = minimum-value	Forces the parameter to be set to at least the specified minimum value. The parameter is set high if AUTOGEN recognizes it warranted
MAX_parameter = maximum-value s	Restricts the parameter from being set higher than the pecified maximum value

18.12.2 AUTOGEN Utility

You should use SYSMAN only to display system parameters; changes should be performed using the AUTOGEN utility. Frequently, a change in one parameter requires changes in related parameters. The AUTOGEN utility allows you to specify parameter changes, and it automatically calculates the required changes to related parameters.

You make parameter adjustments with AUTOGEN by adding or modifying entries in file SYS$SYSTEM:MODPARAMS.DAT. MODPARAM entries have a number of valid formats, as detailed in Table 18.6.

18.12.2.1 System Parameter Entries

To set a parameter to a specific value, specify the parameter name and value separated with an equal sign (=). For example:

```
QUANTUM=10
```

If you are making an incremental change to a parameter, you should use the ADD_format. For example, if you plan to install a new application with the /SHARE option and need 1000 additional global pages, use the following format:

```
ADD_GBLPAGES 1000          ! Installed VAX C 5/1/95
```

By using this type of format, you create an audit trail of parameter changes. If you want to ensure that AUTOGEN establishes a minimum or maximum value for a particular parameter, you can also include an MIN_ and MAX_entry.

18.12.2.2 Page, Swap, and Dump File Entries

In addition to entries that control system parameter values, a number of special AUTOGEN entries control the size of the page, swap, and dump files. Use the following entries to specify the sizes for the dump file and primary page swap files:

```
PAGEFILE = size
SWAPFILE = size
DUMPFILE = size
```

If you want to manage page and swap files manually, you should disable AUTOGEN sizing by adding the following MODPARAM entries:

```
PAGEFILE = 0
SWAPFILE = 0
DUMPFILE = 0
```

To specify the name and size for additional page and swap files, use the following entry formats (where n indicates the page or swap file number):

```
MIN_PAGEFILEn_NAME = file
MIN_PAGEFILEn_SIZE = size
MIN_SWAPFILEn_NAME = file
MIN_SWAPFILEn_SIZE = size
```

18.12.2.3 Establishing Cluster-common Parameters

In the VMScluster environment, you may have several system parameters and should have the same value on each cluster member. To simplify management of these parameters, you can establish a cluster common MODPARAMS file, as follows:

```
$ EDIT SYS$COMMON:[SYSEXE]COMMON-MODPARAMS.DAT

! Cluster Common MODPARAMS.DAT
FREELIM=1000
FREEGOAL=2000
WSMAX=32768
```

Table 18.7 AUTOGEN Phases

AUTOGEN Phase	*Description*
SAVPARAMS	Saves current tuning feedback data to the file AGEN$FEEDBACK.DAT
GETDATA	Gathers information required to calculate proper parameters from MODPARAMS and AGEN$FEEDBACK
GENPARAMS	Generates new system parameter values
TESTFILES	Calculates page, swap, and dump file sizes. Creates AUTOGEN feedback report (AGEN$PARAMS.REPORT)
GENFILES	Creates (or extends) page, swap, and dump files if necessary
SETPARAMS	Copies existing system parameters to VAXVMSSYS.OLD. Then invokes SYSMAN to set parameters
SHUTDOWN	Performs an orderly system shutdown
REBOOT	Performs an automatic system reboot

Then add an INCLUDE statement to each member-specific MODPARAMS as follows:

```
$ EDIT SYS$SPECIFIC:[SYSEXE]MODPARAMS.DAT

!
! Include Cluster-common parameters
!
AGEN$INCLUDE_PARAMS SYS$COMMON:[SYSEXE]COMMON-MODPARAMS.DAT
!
! Member-specific parameters
!
.
.
.
```

18.12.2.4 Invoking AUTOGEN

Once you've added the appropriate entries in MODPARAMS, invoke AUTOGEN as follows:

```
$ @SYS$UPDATE:AUTOGEN start-phase end-phase feedback-mode
```

The valid AUTOGEN phases are listed in Table 18.7.

Table 18.8 AUTOGEN FEEDBACK MODES

Feedback Mode	*Description*
FEEDBACK	Forces AUTOGEN to use tuning feedback data when calculating system parameter values
NOFEEDBACK	Forces AUTOGEN to ignore feedback data
CHECK_FEEDBACK	Instructs AUTOGEN to use feedback data as long as there is at least 24 hours of data and the data is not more than 30 days old

When you're ready to run AUTOGEN you generally start with the SAVPARAMS phase and end with the SETPARAMS phase, as follows:

```
$ @SYS$UPDATE:AUTOGEN SAVPARAMS SETPARAMS
```

You can optionally extend the end phase to include SHUTDOWN or REBOOT if you want AUTOGEN to perform an automatic shutdown or reboot.

18.12.2.5 AUTOGEN Feedback

During normal system operation, OpenVMS maintains statistics regarding utilization of system resources. By default, AUTOGEN uses this data to determine settings for many system parameters. You can override the default usage of feedback data by specifying the feedback mode as detailed in Table 18.8.

By default, AUTOGEN uses feedback data to calculate parameter settings. If you prefer to retain control over when and how system parameters are changed, you should force AUTOGEN to ignore feedback, as follows:

```
$ @SYS$UPDATE:AUTOGEN SAVPARAMS SETPARAMS NOFEEDBACK
```

You can review the parameter changes that feedback data suggests by generating an AUTOGEN feedback report as follows:

```
$ @SYS$UPDATE:AUTOGEN SAVPARAMS TESTFILES FEEDBACK

$ TYPE SYS$SYSTEM:AGEN$PARAMS.REPORT

AUTOGEN Parameter Calculation Report on node: BIRCH
This information was generated at 1-OCT-1995 16:08:23.09
AUTOGEN was run from GENPARAMS to TESTFILES - default execution
specified
```

```
** No changes will be done by AUTOGEN**

The values given in this report are what AUTOGEN would have set the
parameters to.

Feedback information was collected on 26-JUL-1995 15:17:33.62
Old values below are the parameter values at the time of collection.
The feedback data is based on 2 hours of up time.
Feedback information will be used in the subsequent calculations.
Parameter information follows:
------------------------------

MAXPROCESSCNT parameter information:
     Feedback information.
          Old value was 20, New value is 24
          Maximum Observed Processes: 17

Information on VMS executable image processing:
          Processing SYS$MANAGER:VMS$IMAGES_MASTER.DAT

GBLPAGES parameter information:
     Feedback information.
          Old value was 30000, New value is 24000
          Current used GBLPAGES: 12410
          Global buffer requirements: 1024

...

ACP_DINDXCACHE parameter information:
     Feedback information.
          Old value was 9, New value is 9
          Hit percentage: 86%
          Attempt rate: 4 attempts in 10 sec.

ACP_HDRCACHE parameter information:
     Feedback information.
          Old value was 36, New value is 43
          Hit percentage: 34%
          Attempt rate: 13 attempts per 10 sec.

ACP_MAPCACHE parameter information:
     Feedback information.
          Old value was 8, New value is 8
          Hit percentage: 6%
          Attempt rate: 0 attempts per 10 sec.
```

When you perform a system shutdown (as discussed in chapter 6), you can optionally select the SAVE_FEEDBACK shutdown option and force current feedback data be written to the following file:

```
$ SYS$SYSTEM:AGEN$FEEDBACK.DAT
```

To use the old feedback with AUTOGEN, rename the old AGEN$FEEDBACK.DAT file so that it is the most current feedback file. Then run AUTOGEN starting with the GETDATA phase, as follows:

```
$ @SYS$UPDATE:AUTOGEN GETDATA TESTFILES FEEDBACK
```

By skipping the SAVPARAMS phase, you prevent AUTOGEN from writing current feedback statistics to the AGEN$FEEDBACK file.

To implement the changes recommended by the feedback report, execute the following command:

```
$ @SYS$UPDATE:AUTOGEN GETDATA REBOOT FEEDBACK
```

Or, you can manually add the suggested changes to the MODPARAMS file, and run AUTOGEN.

Notes

1. *Guide to OpenVMS Performance Management*, Digital, May 1993, pp. 2–23.
2. Ibid., pp. 3–32.

18.13 Additional References

Coburn, James W., *VMS Performance Management*, Professional Press, 1992.
Ellis, Bruce, *The Hitchhiker's Guide to VMS*, Professional Press, 1990.
Goldenberg, Ruth E., Kenah, Lawrence J., *VAX/VMS Internals and Data Structures (VMS 5.4)*, Digital Press, 1991.

[OpenVMS 5.0]

Guide to VMS Performance Management, Digital, April 1988.

[OpenVMS 6.0]

Guide to OpenVMS Performance Management, Digital, May 1993.
OpenVMS System Manager's Manual: Tuning, Monitoring, and Complex Systems, chapters 14–18, Digital, March 1994.

19 NETWORK MANAGEMENT

Before the early 1980s, most users interacted with Digital systems using directly connected terminals (Figure 19.1). Frequently, systems operated in a stand-alone manner and were not connected to any type of network. To transfer data between systems you had to copy it to magnetic media and physically transport it to another system.

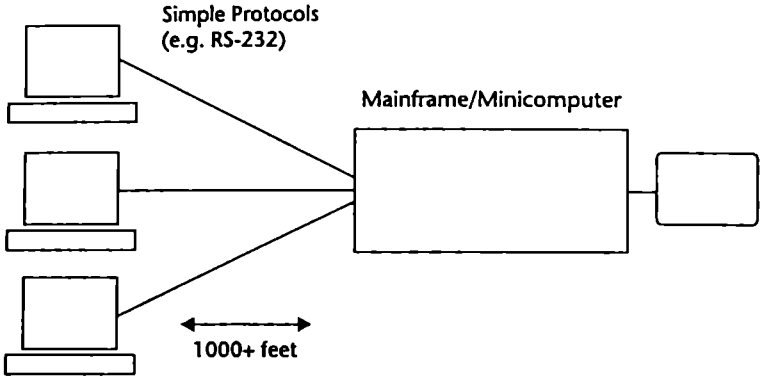
As the installed base of systems and personal computers swelled, transporting magnetic media from host to host ("sneaker-net") became impractical—direct file transfer between systems was desperately needed. Users also needed to access more than one host from a single terminal, and many hosts needed to access common printers. Digital developed the DECnet protocol (and its associated applications) to address the file transfer requirement. Digital later developed the LAT protocol to provide greater flexibility for terminal-based connectivity.

OpenVMS systems can be networked by means of a wide variety of physical connections (e.g., Ethernet, FDDI, Token Ring, serial line). OpenVMS systems can communicate using many other protocols (e.g., TCP/IP, Novell IPX). However, because most Digital users rely on DECnet and LAT protocols on Ethernet, this book focuses that environment. For further information, refer to the additional references section at the end of this chapter.

19.1 Ethernet Basics

Ethernet, developed in the early 1980s, allows up to 1023 nodes to be connected by means of inexpensive coaxial or unshielded twisted-pair (UTP) cabling. With additional devices, Ethernet networks can be extended to support tens of thousands of nodes that span several thousand miles. Ethernet also provides a relatively high transfer speed (a theoretical maximum of 10 Mb/sec). Because of its low cost and good performance, Ethernet has become

Figure 19.1 Directly Wired Terminals



the dominant physical infrastructure for Digital local area networks (LANs). Figure 19.2 shows a sample Ethernet network. Bridges, repeaters, and routers are discussed later in this chapter.

19.1.1 Ethernet Addressing

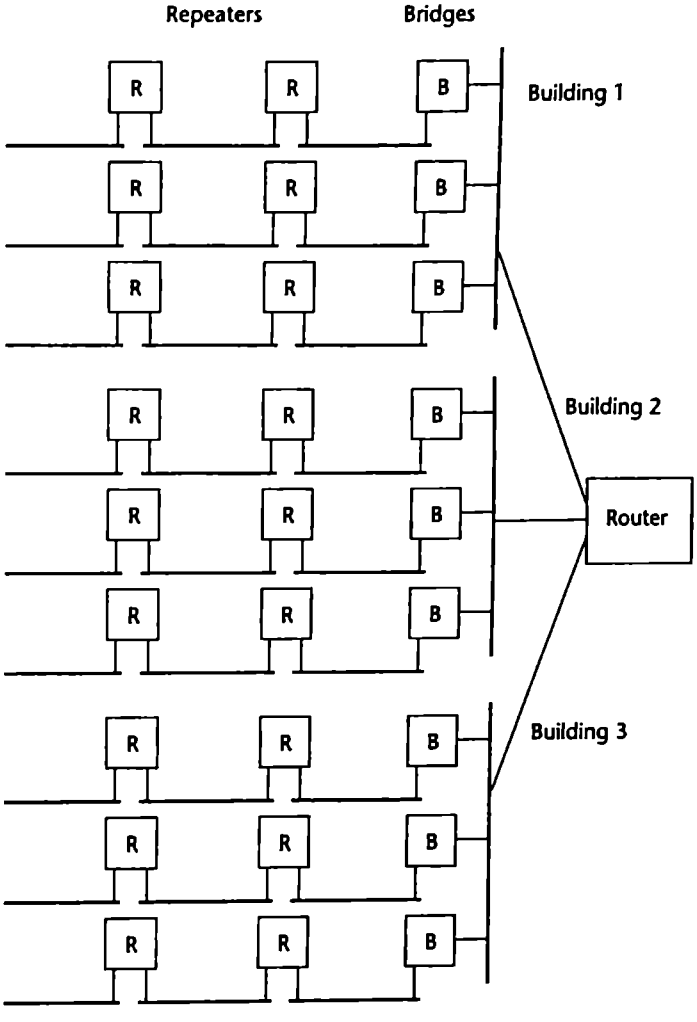
Every node in an Ethernet network has a unique 48-bit address recorded in its adapter. This address is frequently referred to as the physical, datalink, or media-access control (MAC) address of a node. Whenever a node transmits data onto an Ethernet network, it places the data in an Ethernet frame. The beginning of the frame contains the destination *datalink* address of the target node and the datalink address of the source node, followed by the data, and finally ending in a frame checksum, as follows:

```
Ethernet Header
AA-00-04-00-02-14        [Destination Datalink Address: Bytes 0 - 5]
AA-00-04-00-01-04        [Source Datalink Address: Bytes 6 - 11]
60-03                    [Protocol Type: Bytes 12-13]

Ethernet Data
...
Data                     [Data: 46 - 1500 Bytes]
...

Ethernet Trailer
00-00-00-00              [Checksum: 4 Bytes]
```

Figure 19.2 Example of an Ethernet Network



The first 6 bytes are the destination datalink address (dashes are added between each byte for readability). The next 6 bytes are the source datalink address, a 2-byte protocol type, then between 46 and 1500 bytes of data, and finally a 4-byte packet checksum at the end of the frame. An Ethernet frame must contain a minimum of 64 bytes and a maximum of 1518 bytes. Because Ethernet frames are used by many different protocols, the protocol type field helps easily identify which protocol is using the frame. Some common protocol types (in hexadecimal) are as follows:

Table 19.1 Ethernet Cable Types

Cable Type	*Maximum Segment Length (m)*	*Maximum Attachments per Segment*
Unshielded twisted pair (UTP)	100	1
Thinwire coaxial (RG58/U)	180	31
Thickwire coaxial	500	100

```
60-03     LAT
60-04     DECnet
80-00     TCP/IP
80-9B     Appletalk
81-37     Novell
```

19.1.2 Ethernet Terminology

Networking terms such as *node*, *segment*, and *LAN* are frequently used to describe two entirely different entities. The following section defines these terms as they should be interpreted within the context of this chapter.

Node

A *node* is an OpenVMS host, terminal server, personal computer (PC), or other network device that has an Ethernet datalink address. When a node transmits a frame, all other nodes on the same LAN "hear" the packet. However, an Ethernet adapter passes the contents of the frame to higher level software only if the destination datalink address of the frame matches the adapter's datalink address.

Physical Segment

A *physical segment* is a length of cable with one or more attached nodes. Depending on the type of cable used, segment lengths and maximum attachments are limited as indicated in Table 19.1.

The length limits exist because as the signal travels from one end of a segment to another, it becomes weaker and begins to distort.

Repeater

A repeater is a network device with two or more network interfaces. When attached to two physical segments, a repeater listens to the signal on one segment and immediately retransmits it (one bit at a time) on all its other

interfaces. As a repeater retransmits, it re-amplifies and re-times the signal so that the signal is strong enough to travel the length of the next segment. Repeaters are *not* nodes, they do not have a datalink address.

A series of physical segments connected by repeaters creates a single *logical segment*, also called a *local area network* (LAN). Ethernet specifications allow a maximum of four repeaters between any two network stations within a LAN. Thus, the maximum diameter of an Ethernet LAN is 2500 m (five thickwire segments connected with four repeaters).

Bridge

The diameter of a LAN is restricted so that a packet can propagate from one side of a LAN to the other in a reasonable amount of time. A bridge is network device with two or more network interfaces. You use bridges to connect several LANs to create a single *extended LAN.*

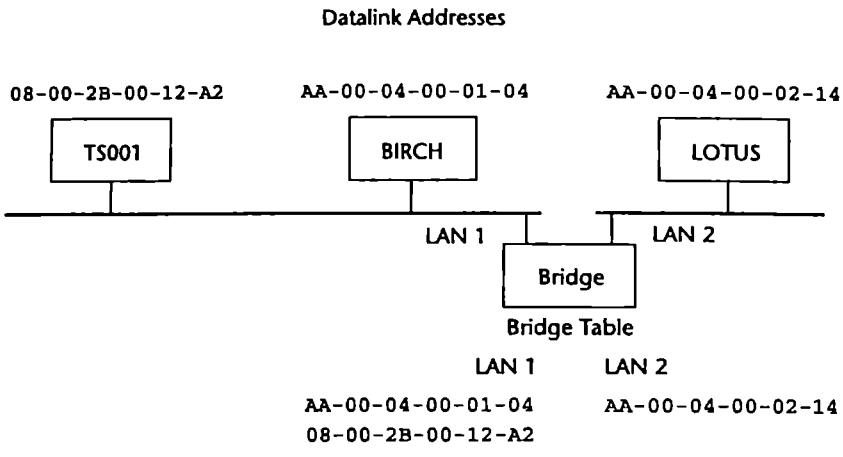
A bridge has a datalink address for each Ethernet interface it contains. When a bridge detects the start of a packet on one of its interfaces, it reads the *entire packet* and stores it in a buffer. Only after it has received the entire packet does it then forward the packet to all its other interfaces. Although bridges are full Ethernet nodes (they have a datalink address), packets are usually not sent directly to them. Rather, bridges operate in *promiscuous mode*, which allows them to receive and process all packets on a LAN—even when the destination address of a packet does not match the address of the bridge.

Most Ethernet bridges are classified as *transparent learning bridges. Learning* because as a bridge receives packets it examines the source address of each packet and records the interface on which the packet was received. *Transparent* because bridges receive and forward packets without changing them in any way (see Fig. 19.3).

Once a bridge has learned, it filters traffic destined for the same LAN from which it was received. For example, if TS001 sends a packet to LOTUS, the bridge receives it from LAN1 and forwards it to LAN2. However, if TS001 sends a packet to BIRCH, the bridge receives the packet on LAN1, but it notices that its bridge table indicates that BIRCH is connected to LAN1. Thus, the bridge assumes that the packet has already reached its destination and discards the packet. The end result is that any traffic between TS001 and BIRCH does *not* propagate to LAN2. Although bridges are necessary to extend a LAN, they can also be used to segment traffic.

Many bridges support Ethernet and serial interfaces. Using serial interfaces, modems, and leased phone lines, you can create extended LANs that cover many thousands of miles.

Figure 19.3 Example of a Learning Bridge



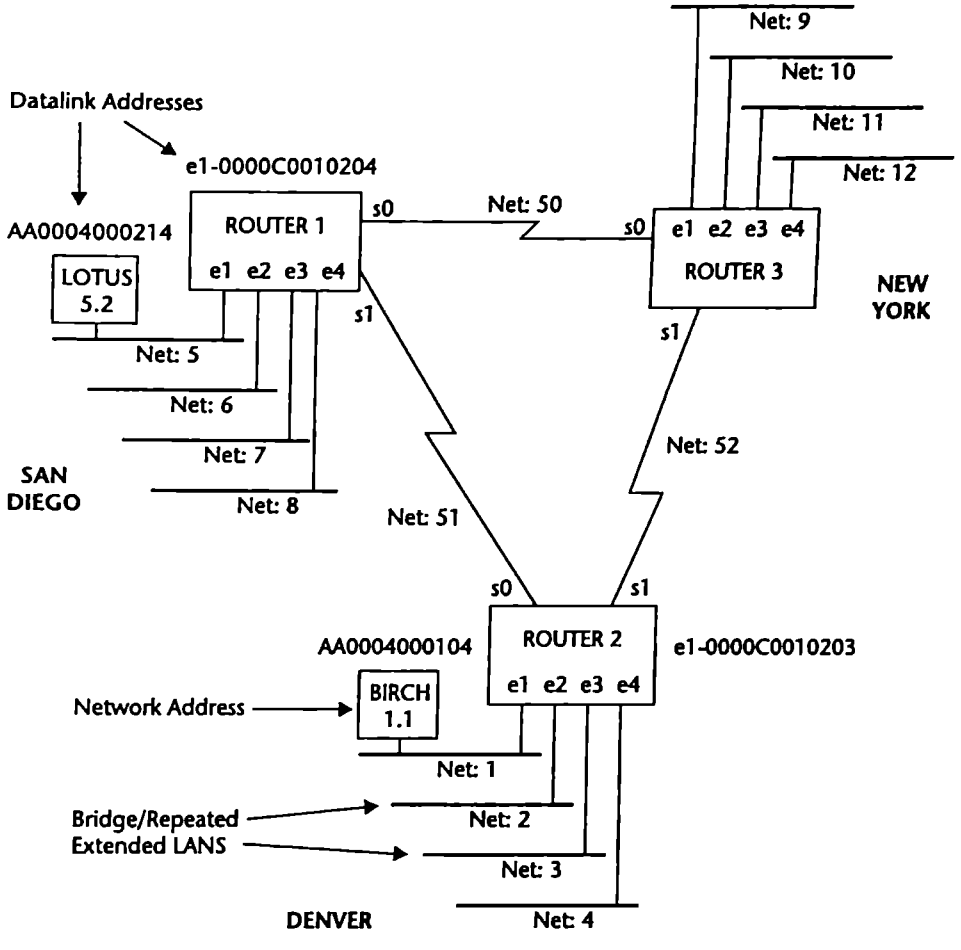
Router

As the number of nodes in a LAN increases dramatically, bridges become impractical because bridge tables grow excessively large. Bridges must keep track of every datalink address in an entire extended LAN (not just the LANs to which they are attached). If you were to depend on bridges to build a large network, you could easily have bridge tables containing tens of thousands of entries.

Routers resolve this problem by using a second layer of node addressing—a *network address*. Network address formats are different for every protocol, but they usually consist of two components: a *network number* and a *node number*. Typically, each LAN (or extended LAN) is assigned a unique network number and each node on that LAN is assigned a unique node number. Routers are then used to interconnect multiple networks to build an *internetwork*. Routers generally only have to keep track of each network within the internetwork (not every node). An example internetwork is shown in Figure 19.4.

In Figure 19.4, the internetwork consists of 12 extended LANs (Networks 1–12) connected by three routers. Each router has four Ethernet interfaces (e1–e4) and two serial interfaces (s0, s1). Unlike configuring a bridge, configuring a router is done by telling it which network numbers it is attached to. After a router begins operating, it periodically transmits routing update packets describing the networks about which it knows (the networks to which it is directly attached). Other routers listen to these updates and add the information to their routing tables. As routers listen to routing updates, and send

Figure 19.4 Example of a Routed Internetwork



their own updates, all routers eventually build tables that detail the topology of the entire internetwork. Table 19.2 shows an example of routing tables for ROUTER 1.

Routers may receive routing updates describing several possible paths to the same network. In this case routers use the information that contains the lowest hop count. *Hop count* represents the number of routers that must be traversed to reach a given network.

DECnet uses a routing scheme similar to the one just described. It is described in more detail in the next section. LAT was designed to work within a LAN or extended LAN only. The main objectives were to deliver quick re-

Table 19.2 ROUTER1 Routing Tables

Network Number	*Hop Count*	*Via Interface*	*Learned From*
1, 2, 3, 4	1	s1	ROUTER 2
5, 6, 7, 8	0	e1, e2, e3, e4	Self
9, 10, 11, 12	1	s0	ROUTER 3
50	0	s0	Self
51	0	s1	Self
52	1	s1	ROUTER 2

sponse time (critical for interactive terminal sessions) with minimal overhead. To meet these objectives, Digital chose not to add a network layer to the LAT protocol. As a result, LAT cannot be routed—it can only be bridged (or repeated).

19.2 DECnet Introduction

Initially, Digital developed the DECnet protocol to allow file transfers between Digital hosts. The early versions of DECnet (Phase I through Phase III) allowed low speed (e.g., 56 Kb/sec), point-to-point connections. DECnet (Phase IV), introduced in the early 1980s, enhanced networking capabilities by allowing Digital hosts to participate in Ethernet-based LANs.

DECnet is a communication protocol used by several DECnet applications. Some of the more popular DECnet applications are listed in Table 19.3.

19.3 DECnet Components

19.3.1 DECnet Addressing

DECnet is designed to operate in a LAN or an internetwork (multiple LANs separated by routers). Like other Ethernet nodes, DECnet nodes address each other (within an extended LAN) by means of datalink addressing. To support internetworking, DECnet nodes also use a 16-bit network address. The first 6 bits represent the *area* number (1 through 63) and the last 10 bits represent the host number (1 through 1023). The area number identifies a distinct extended LAN, and the host number identifies a host within a LAN. As a result, DECnet allows a maximum of 64,449 (63*1023) nodes to participate in a DECnet internetwork.

Table 19.3 DECnet Applications

Application	*Description*
Remote file access	All DCL file-manipulation commands (COPY, CREATE, DELETE, RENAME, etc.) can be used throughout the network. However, accessing remote files is slightly different from accessing local files. To access remote files you must explicitly specify the remote node name. File access can also be much slower for remote files, especially for large files
Remote login	An interactive login can be initiated from any node to any node in the network using the DCL SET HOST command
Remote job submission	You can submit print or batch jobs to execute or print on a remote system. However, the print or batch job file must first be copied manually to the remote node before it can be submitted
Network-wide mail	You can use the VMS Mail utility to send mail to a user on a remote node by preceding the username with the remote DECnet node name

DECnet networks consist of three types of nodes: end nodes, Level 1 routers, and Level 2 routers.

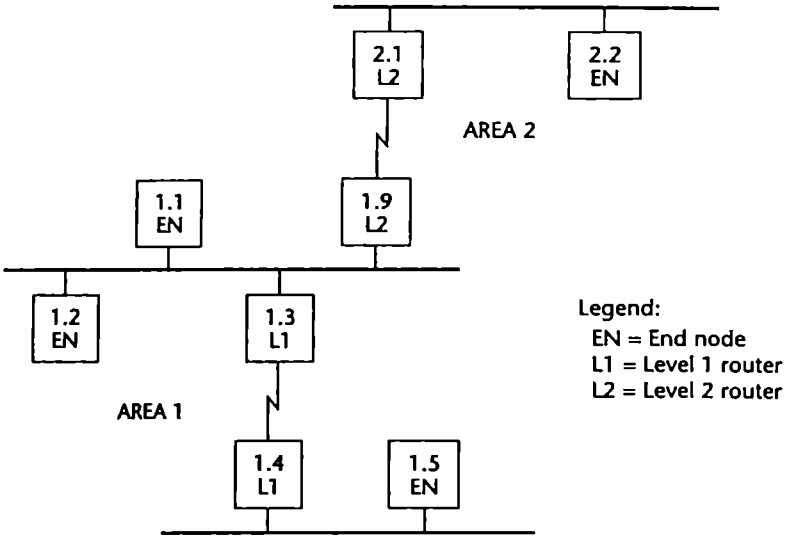
19.3.2 End Nodes

End nodes are allowed to have one active network connection. End nodes can have multiple Ethernet adapters or serial DECnet links, but only one can be started at any given time. An end node can communicate directly only with nodes to which it is directly connected (e.g., on the same Ethernet). In most cases the other nodes must also be in the same area. End nodes periodically transmit "hello" messages so that Level 1 routers (see section 19.9.3) know they are available. End nodes ignore hello messages transmitted by other nodes and thus do not keep track of the availability of other end nodes.

19.3.3 Level 1 Routers

Level 1 (L1) Routers are responsible for tracking the availability of all end nodes within the areas to which they are attached. In Figure 19.5, nodes 1.1, 1.2, and 1.9 transmit end node hello messages that are received by L1 router 1.3. Node 1.3 adds those nodes to its routing tables. L1 routers periodically transmit L1 router hello messages so that end nodes can select a designated L1 router. End nodes select an L1 router by selecting the hello message con-

Figure 19.5 Example of a DECnet Internetwork



taining the highest router priority. In case of a tie, the router with the highest DECnet node number wins. In Figure 19.5, node 1.3 is the designated L1 router for nodes 1.1 and 1.2.

Every few minutes, L1 routers within an area exchange L1 routing tables. The routing tables describe each end node, the number of hops (routers to traverse) to reach the end node, and a path cost. *Path cost* is a routing metric that more accurately describes the shortest path between two points. Two paths might exist to the same destination with a hop count of 2. However, one path might consist of two Ethernet hops and the other of two 19,200-baud serial links. In a DECnet network, an Ethernet circuit is typically assigned a cost of 4, and a 19,200-baud serial connection is assigned a cost of 10. In the foregoing example, the Ethernet path would have a total cost of 8, as opposed to a cost of 20 for the serial links. By choosing the least cost path, DECnet tries to assure that the best path is used. When L1 routers receive updates from other L1 routers, they update their routing tables to include the least cost path to each end node and the address of the routers from which they learned the route. Eventually, every L1 router knows the least cost path to every end node within its area.

When an end node wants to send data to another end node *within the same area*, it sends the data to its designated L1 router. The L1 router then checks the availability of the target end node. If the end node is reachable, the L1 router forwards the data to the end node or to the next L1 router in the path to the end node. For example, if end node 1.1 were to send data to end node 1.2, it would first forward the data to the designated L1 router (node 1.3). The L1 router would see that node 1.2 was available and directly accessible—it would thus forward the data directly to node 1.2. In this case the router would have received and forwarded the data on the same Ethernet interface. When this occurs, the router sets a special bit in the frame, known as the *intra-Ethernet bit*.

The intra-Ethernet bit is used to signal the receiving end node that it is attached to the same LAN as the original source node. From this point on, the receiving node knows that it can communicate with the source node directly, without the assistance of a router. On the other hand, if end node 1.2 needed to communicate with end node 1.5, *all* end nodes would be routed through the L1 router node 1.3.

Digital hosts configured as L1 routers can also perform all end node functions.

19.3.4 Level 2 Routers

Level 2 (L2) routers are responsible for tracking the availability and path to all areas (networks) within an entire DECnet internetwork. In Figure 19.5, for example, node 2.1 knows that it is directly attached to area 2. It adds an entry in its L2 routing tables indicating that area 2 is zero hops away. Periodically, node 2.1 exchanges routing tables with other L2 routers with which it is directly attached (e.g., node 1.9). When node 1.9 receives a routing update from node 2.1, it adds information concerning area 2. Similarly node 2.1 learns about area 1 from node 1.9.

When an end node forwards traffic to an L1 router and the router determines that the data is destined for a different area, the L1 router forwards the data to the designated L2 router for that area. The L2 router then forwards the data to the next hop in the path to the final destination. This process continues until the data reaches its destination LAN.

OpenVMS systems configured as L2 routers also perform all L1 and end node functions.

19.4 DECnet Management

You manage a DECnet node using three major utilities: NETCONFIG.COM, STARTNET.COM, and the Network Control Program (NCP). Use NETCONFIG to configure a DECnet node, STARTNET to start DECnet, and NCP to stop DECnet and to perform other management functions.

Invoke NCP using the following DCL commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>
```

These utilities are discussed in more detail in the following sections.

19.4.1 Assigning Addresses and Host Names

Every DECnet host must have a unique network address and host name. Initially, this limitation was not an issue for most Digital customers because they had far less than 64000 systems. However, if you integrate PCs into a DECnet network by means of Digital's Pathworks product, you can very easily use all DECnet addresses. As a result, assign DECnet addresses sparingly.

You should try to assign only one area number for each extended LAN. Otherwise, you will need to add area routers to route between the two areas, even though the nodes are in the same extended LAN.

DECnet host names consist of an alpha-numeric string of up to six characters, such as BIRCH, LOTUS. You can choose host names that have geographic significance (e.g., LA, NYC, PHIL), but do this only if there is little chance that nodes will move.

19.4.2 Configuring a DECnet Node

Once you have chosen a DECnet network address and host name, you are ready to configure DECnet using the NETCONFIG utility. Log in to the node you wish to configure and proceed as follows:

```
$ @SYS$MANAGER:NETCONFIG

    DECnet for OpenVMS network configuration procedure

This procedure will help you define the parameters needed to get
DECnet running on this machine. You will be shown the changes before
they are executed, in case you wish to perform them manually.
```

```
What do you want your DECnet node name to be? : BRO
What do you want your DECnet address to be? : 4.1
Do you want to operate as a router? [NO (nonrouting)]: NO
```

NO configures the node as an end node.
YES configures the node as a router.

```
The network object database file is
SYS$SYSROOT:[SYSEXE]NETOBJECT.DAT;1.

Do you want to purge the object database? [YES]: NO
Do you want a default DECnet account? [NO]: [RETURN]
```

The foregoing procedure allows users on remote nodes to copy files to and from the default directory for the DECNET account on this system. By enabling this account, remote users could consume excessive disk space on this node.

```
Do you want a default account for the MAIL object? [YES]: NO
Do you want a default account for the FAL object? [NO]: [RETURN]
```

If you respond YES, then nonprivileged users can access the FAL$SERVER (file access listener) directory from a remote node. For example

```
PEG$ DIR BRO::

Directory BRO::SYS$SPECIFIC:[FAL$SERVER]

NETSERVER.LOG;1

Total of 1 file.
PEG$
```

Without the FAL$SERVER account, remote users get the following error message:

```
PEG$ DIR BRO::
%DIRECT-E-OPENIN, error opening BRO::*.*;* as input
-RMS-E-FND, ACP file or directory lookup failed
-SYSTEM-F-INVLOGIN, login information invalid at remote node

Do you want a default account for the PHONE object? [YES]: NO
```

The foregoing procedure allows users on remote nodes to contact users on this node using the PHONE utility. See your OpenVMS documentation for more information on the PHONE utility.

```
Do you want a default account for the NML object? [YES]: NO
```

This option enables or disables the network management listener (NML) object. By enabling the NML object on this node, non-privileged users on remote nodes can use the NCP TELL command to view network configuration parameters and counters.

With the NML listener, a remote user can use the TELL command, as follows:

```
NCP> TELL BRO SHOW KNOWN NODES

Known Node Volatile Summary as of 26-MAY-1994 21:15:55

Executor node = 4.5 (BRO)

State          = on
Identification = DECnet-VAX V5.5-2, VMS V5.5-2

Node           State     Active     Delay     Circuit    Next node
                         Links
4.1 (FRED)                                    SVA-0      4.2 (LOTUS)
4.2 (LOTUS)                                   SVA-0      4.2 (LOTUS)
```

Without NML, you would get the following error message:

```
FROM node PEG
NCP> TELL LOTUS SHOW KNOWN NODES
%NCP-F-CONNEC, unable to connect to listener
-SYSTEM-F-INVLOGIN, login information invalid at remote node

Do you want a default account for the MIRROR object? [YES]: NO
```

This option enables or disables the MIRROR object. When the MIRROR object is enabled, nonprivileged users on remote nodes can perform a loopback test to this node. For example, with the MIRROR object enabled, a user on node PEG could perform a successful loop test (e.g., no error message is returned).

```
NCP> LOOP NODE BRO
```

Without the MIRROR object, the following error message would be generated:

```
PEG$ MC NCP
NCP> LOOP NODE BRO
%NCP-W-MIRCTF, Mirror connect failed , Access control rejected
Messages not looped = 1

Do you want a default account for the VPM object? [YES]: NO
```

The VPM (Vax Performance Monitor) object enables users on remote nodes (within a VMScluster) to display performance statistics on this node. This object is typically used when you issue the DCL MONITOR/NODES=(node1, node2, etc.) or the MONITOR/CLUSTER command.

```
Here are the commands necessary to setup your system.

PURGE KNOWN CIRCUITS ALL
PURGE KNOWN LOGGING ALL
PURGE MODULE CONFIGURATOR KNOWN CIRCUITS ALL
$ DEFINE/USER SYS$OUTPUT NL:
$ DEFINE/USER SYS$ERROR NL:
$ RUN SYS$SYSTEM:NCP          ! Remove existing entry, if any
     PURGE NODE 4.1 ALL
     PURGE NODE FRED ALL
$ RUN SYS$SYSTEM:NCP
     DEFINE EXECUTOR ADDRESS 4.1
     DEFINE EXECUTOR STATE ON
     DEFINE EXECUTOR NAME FRED
     DEFINE EXECUTOR MAXIMUM ADDRESS 1023
     DEFINE EXECUTOR TYPE NONROUTING IV
     DEFINE OBJECT TASK NUMBER 0 USER ILLEGAL PASSWORD DISABLED
     DEFINE LINE SVA-0 STATE ON
     DEFINE CIRCUIT SVA-0 STATE ON COST 4
     DEFINE LOGGING MONITOR STATE ON
     DEFINE LOGGING MONITOR EVENTS 0.0-9
     DEFINE LOGGING MONITOR EVENTS 2.0-1
     DEFINE LOGGING MONITOR EVENTS 4.2-13,15-16,18-20
     DEFINE LOGGING MONITOR EVENTS 5.0-18
     DEFINE LOGGING MONITOR EVENTS 128.0-4

Do you want these commands to be executed? [YES]: [RETURN]

The changes have been made.

If you have not already registered the DECnet-VAX key, then do so now.

After the key has been registered, you should invoke the procedure
SYS$MANAGER:STARTNET.COM to startup DECnet-VAX with these changes.
```

```
(If the key is already registered) Do you want DECnet started? [YES]:
[RETURN]
%RUN-S-PROC_ID, identification of created process is 00000097
%RUN-S-PROC_ID, identification of created process is 00000099
$
```

19.4.3 Starting DECnet

If you want DECnet automatically started during a system boot, you should add the following command to your system startup command procedures:

```
$ @SYS$MANAGER:STARTNET
```

While DECnet is starting, messages similar to the following will be displayed:

```
%RUN-S-PROC_ID, identification of created process is 00000092
%%%%%%%%%%% OPCOM 14-APR-1994 13:05:52.95 %%%%%%%%%%%
Message from user DECNET
DECnet starting

%%%%%%%%%%% OPCOM 14-APR-1994 13:06:00.48 %%%%%%%%%%%
Message from user DECNET
DECnet event 4.10, circuit up
From node 4.1 (BIRCH), 14-APR-1994 13:05:59.23
Circuit SVA-0

%RUN-S-PROC_ID, identification of created process is 00000094
```

19.4.4 Shutting Down DECnet

To manually shut down DECnet use the following NCP command:

```
$ RUN SYS$SYSTEM:NCP
NCP> SET EXECUTOR STATE OFF
```

Before you shut down DECnet, you should use the NCP command SHOW KNOWN LINKS to display active DECnet links. This tells you if remote users have active SET HOST or file transfer connections to the host you want to shut down.

19.4.5 Basic DECnet Troubleshooting

If you are having trouble getting two DECnet nodes to communicate, perform the following steps:

1. Identify designated L1 router. Log in to one of the end nodes and use the DCL SHOW NETWORK command to determine the designated L1 router, as follows:

```
BRO$ SHOW NETWORK
VAX/VMS Network status for local node 4.5 BRO on 26-MAY-1994   20:45:58.60

This is a nonrouting node, and does not have any network information.
The designated router for BRO is node 4.2 LOTUS
```

2. Examine L1 routing tables. Log in to the designated L1 router and examine its L1 routing tables, as follows:

```
LOTUS$ SHOW NETWORK
VAX/VMS Network status for local node 4.2 LOTUS on 4-FEB-1994  08:52:31.14

The next hop to the nearest area router is node 4.1 FRED.

Level 1 Routing Tables

Node            Links    Cost  Hops     Next Hop to Node
4.2   LOTUS       0        0     0       (Local)    ->    4.2   LOTUS
4.1   FRED        0        4     1       SVA-0      ->    4.1   FRED
4.5   BRO         0        4     1       SVA-0      ->    4.5   BRO

Total of 3 nodes.
```

If the target node is in the same area, it should be listed in the foregoing table. If it is not, then the L1 router is not receiving the hello from the target end node.

If the target end node is in a different area, determine the designated L2 router from the output. In this example, node 4.1 (FRED) is the designated L2 router.

3. Examine routing tables on designated L2 router. Log in to the designated L2 router and issue the following command:

```
FRED$ SHOW NETWORK
VAX/VMS Network status for local node 4.1 FRED on 26-MAY-1994
20:56:22.85

Level 2 Routing Tables

Area    Cost    Hops    Next Hop to Area
4        0       0      (Local)    ->    4.1   FRED
5       10       1      TT-0-1     ->    5.1   PEG


Level 1 Routing Tables

Node    Links      Cost   Hops    Next Hop to Node
4.2     LOTUS       0      4      1  SVA-0   ->    4.2   LOTUS
4.5     BRO         0      4      1  SVA-0   ->    4.5   BRO

Total of 2 nodes.
```

4. Trace L2 routing tables. Identify the next L2 router in the path to the target node (e.g., PEG). Log in to that node, and continue to trace the path until you reach an L2 router in the target area. Frequently, the problem is that a link is down and the target area is listed as unreachable.

```
PEG$ SHOW NETWORK

VAX/VMS Network status for local node 5.1 PEG on 26-MAY-1994
21:03:07.44

Level 2 Routing Tables

Area    Cost       Hops       Next Hop to Area
4        10         1         TT-0-1 -> 4.1 FRED
5         0         0         (Local) -> 5.1 PEG


Level 1 Routing Tables

Node          Links     Cost   Hops       Next Hop to Node
5.1 PEG       0          0      0         (Local) -> 5.1 PEG
```

Once you reach the target area, examine the L1 routing tables of a router in that area. Finally, trace the path to the target end node.

As you trace the path in one direction, you usually stop at the point where the problem is occurring (e.g., a link is down). If the problem is not obvious,

try tracing the path in the reverse direction. Ideally, you should use this tracing procedure to map out the entire network *before* you have a problem. This way, when a problem occurs, you will be able to pinpoint it more easily.

19.5 Managing LAT (Local Area Transport)

To address the need for more flexible terminal and printer connectivity, Digital developed the LAT protocol and several models of terminal servers (e.g., DECserver 100, 200, 500, 700). A terminal server typically has eight or more serial ports and one Ethernet port. In a LAT network, terminals and printers are connected to terminal servers rather than being directly wired to a specific host. The Ethernet port on a terminal server is then connected to a LAN. Once the system is configured, a user can log in to any LAT host connected to the same extended LAN as the terminal server. Hosts can also direct print queues to printers attached to any terminal server on the same extended LAN.

For example, in Figure 19.6, users attached to terminal servers TS001 and TS002 can log in to either node LOTUS or node BIRCH. Print queues on BIRCH and LOTUS can be directed to printers attached to either TS001 or TS002.

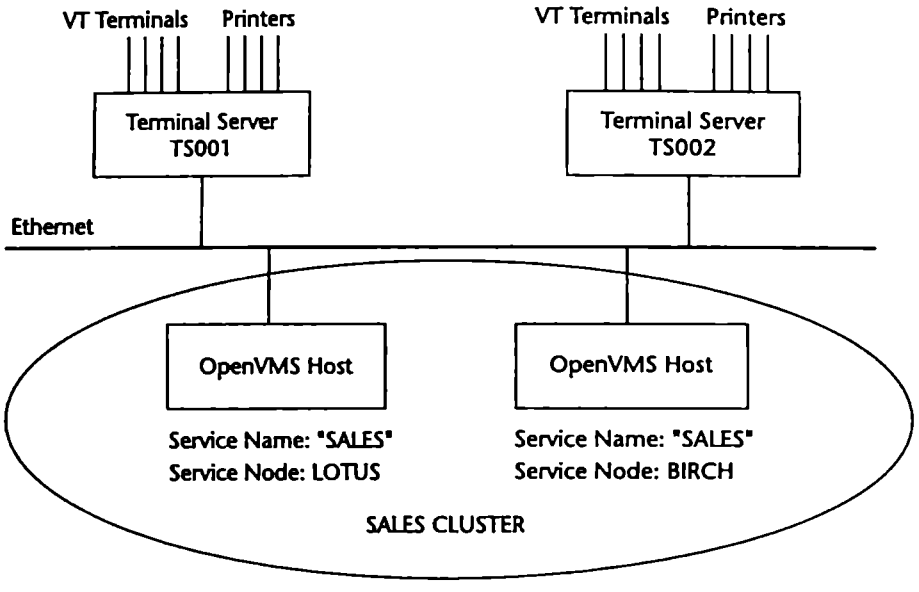
19.5.1 LAT Components

The following three main entities participate in a LAT network.

1. Service Nodes. *Service nodes* offer one or more *service names* to the network. In Figure 19.6, LOTUS and BIRCH are service nodes. Frequently, the service node name and service name are the same. However, they are two distinct separate entities.
2. Service Name. Service names represent services available on a service node. Usually, the service is an interactive login offered by one or more OpenVMS nodes.
3. Terminal Server. Terminal servers (e.g., TS001 and TS002) allow users to initiate interactive terminal sessions to service names. Service nodes also can request that terminal servers establish a connection between a printer port and an OpenVMS print queue.

After LAT is configured and started on an OpenVMS service node, the service node periodically transmits a special packet, called a *service an-*

Figure 19.6 Example of a LAT Network



nouncement, onto the network. Service announcements are sent to a special datalink address (LAT multicast) that causes the packet to be processed by all terminal servers on the LAN. Service announcements contain the service node name, a list of services names being offered, and a numeric rating for each service. The following shows a service announcement sent by service node BIRCH offering the service SALES:

```
Ethernet Header
  Destination:            09:00:2b:00:00:0f LAT Multicast [0-5]
  Source:                 08:00:2b:13:97:2c [6-11]
  Protocol Type:          0x 6004 DEC LAT [12-13]
DEC LAT - Local Area Transport
  Message Type:           10 [14 Mask 0xfc] Service
  Flags:                  0 [14 Mask 0x03] From host No Response
...
...[irrelevant packet contents omitted]
...
Enabled Groups:           %00000001 [27] ! Only Group 0 enabled
  Node Name:              BIRCH [28-32]
  Node Description:       Welcome to VAX/VMS V6.1 [33-61]
  Number of Services:     1 [62]
  Rating:                 83 [63]
```

```
  Service Name:            SALES [64-69]
  Service Description:     Welcome to VAX/VMS V6.1 [70-98]
  Number of Classes:       1 [99]
  Service:                 1 [100] Remote Terminal Service
Frame Check Sequence:      0x40f3de00 [101-104]
```

Terminal servers constantly listen for service announcements and build a table of known service nodes, service names, and ratings. Users cannot connect to a LAT service until the terminal server to which they are connected receives a service announcement.

You can view the service table of a terminal server by entering the following command on the server:

```
TS001> SHOW SERVICE ALL

Service SALES - Available

Node Name       Status          Rating    Identification

BIRCH           Reachable       83        Welcome to VAX/VMS V6.1
LOTUS           Reachable       83        Welcome to VAX/VMS V5.5-2
```

The foregoing output confirms that the terminal server TS001 knows about the service SALES and that it is offered by service nodes BIRCH and LOTUS. Currently, both BIRCH and LOTUS are advertising a service rating of 83, indicating that they are both equally available.

19.5.2 Connection Basics

Once services are known to a terminal server, users can request a connection, as follows:

```
TS001> CONNECT SALES

Local -051- Session 1 [SALES] established

      Welcome to VAX/VMS V6.1 (BIRCH)

Username:
```

When TS001 received a connection request, it looked in its tables and located the node offering the highest rating for the SALES service. In the foregoing example, the server chose BIRCH—in case of a tie, the server picks the first node.

19.5.3 Managing Terminal Servers

The following sections deal with the issues of loading terminal server software, accessing the server console port, and configuring various server parameters. The exact syntax of terminal server commands varies for each server model. For example, some servers do not support the CHANGE command; instead you must use the DEFINE and SET commands to change the permanent and volatile server settings. Refer to your terminal server user's guide for more information on exact command syntax.

19.5.3.1 Loading Server Software

A terminal server is a small computer that runs LAT software. Some terminal servers have LAT software built into read-only memory (ROM) or flash memory. Others must down-line load their software from a host before they can begin operating. If your server requires a down-line load you must install a terminal server load image file on a host. Older Digital terminal servers perform down-line loads from an OpenVMS host using a maintenance operation protocol (MOP). Refer to your terminal server manual for more information on configuring a MOP server, if necessary.

A host does not have to be a MOP server to accept LAT connections. If your terminal servers require MOP downloads, you should configure at least two MOP servers for your network. Otherwise, if your MOP server is down and a terminal server attempts to boot, it will not be able to load its required software. If you are particularly concerned about system performance, you should choose your MOP servers carefully. If you configure an important host to act also as a MOP server, performance could deteriorate after a wide-scale power outage. When power is restored, hundreds of terminal servers could simultaneously issue MOP load requests and inundate your MOP server.

Because of these management issues, the trend is to use terminal servers with built-in software. Servers with flash memory are the most desirable because you can easily perform upgrades by inserting a new flash card. Servers without flash require that you physically replace a ROM chip on the motherboard of your server.

19.5.3.2 Connecting to the Terminal Server Console Port

After a server is installed and loaded, perform management functions by attaching a terminal to one of its ports. If possible you should connect the terminal to port 1. Port 1 is usually the designated console port, and important diagnostic messages are sometimes sent to that port. If the terminal serv-

er is at a remote location, you can connect to the server from a DECnet host using the MOP remote console facility, as follows:

```
[log in to a DECnet host on the same extended LAN as the terminal
server]

$ RUN SYS$SYSTEM:NCP
NCP> SHOW KNOWN CIRCUITS

Known Circuit Volatile Summary as of 14-APR-1994 14:48:52

Circuit          State      Loopback Name  Adjacent Routing Node
SVA-0            on                        4.5
```

Locate the Ethernet circuit attached to the same extended LAN as the remote terminal server. Use that circuit name and the server datalink address in the following command:

```
NCP> CONNECT VIA SVA-0 PHYSICAL ADDRESS 00-80-64-06-01-E2
Console connected (press CTRL/D when finished)

[press RETURN a few times]

WN-5104 Terminal Server V4.212
64B Seaside rm: 202

Enter username: BOGUS          ! type anything

Please type HELP for assistance
WN5104 5>
```

The datalink address of the terminal server should be printed on the back of the terminal server. If you don't know its address, you'll have to have someone look at the server. Digital's servers usually start with *08-00-2B*. In the foregoing example *00-80-64-06-01-E2* is for a Wyse terminal server.

Once connected to the terminal server, you can view its configuration as follows:

```
WN5104 5> SHOW SERVER

WN-5104 Terminal Server V4.212      Uptime:        00:47:18

Ethernet:     00-80-64-06-01-e2     Internet:    192.4.4.1
Name:                   WN5104      Number:              0

Identification:            64B      Seaside rm:        202
```

```
Console Port:          1          Prompt:          "%N %p>%P "
Inactivity Timer:     30          Password limit:            3
Monitor Timer:         3          Session limit:            32
Circuit Timer:        80          Queue limit:             100
Keepalive Timer:      20          Node limit:              200
Multicast Timer:      30          Retransmit limit:          8

Service Groups:        0
Startup File:                                  WN-5104.CFG

Enabled Characteristics:
  Broadcast, Lock, Heartbeat, Announcements
```

19.5.3.3 Initial Terminal Server Configuration

Although there are many terminal server parameters that you can configure, the following are the most important.

Server Name

By default, most Digital terminal servers are named *LAT_xxxxxxxxxxx* where *xxxxxxxxxxx* is the datalink address of the server. Because this naming system is rather cumbersome, you should consider changing the server name to something more meaningful and easier to type. Remember, you will probably reference this name in many places (e.g., in startup files that create LAT device mappings for printers). Some system managers use server names that indicate a geographic location. Don't do that, however, unless you're sure you will never need to relocate the server. Otherwise, you'll have to change all your system startup procedures. An arbitrary naming scheme is perfectly acceptable, for example TS0001, TS0002, etc. (TS for terminal server). A sample syntax is as follows:

```
> CHANGE SERVER NAME "TS0001"
```

Server Passwords

The following two passwords control access to a terminal server:

1. Access Password. Users must enter the access password when passwords are enabled on a specific port (this is sometimes used for modem ports). More important, on most servers you are prompted for the access password when you connect to a server using the MOP remote console facility. By default, most servers are configured with the default access password ACCESS. Thus, by default, any user with access to DECnet can

connect to the remote console port of any terminal server in your network. If this is undesirable, change the access password on all your servers.

2. Privilege Password. To change server parameters, you must first enter the SET PRIVILEGE command followed by the privilege password. By default, nearly all terminal servers use a privilege password of SYSTEM. Unless you change the default privilege password, any user in the network can reconfigure your servers.

The procedures for changing the privilege and access passwords are discussed in chapter 17.

Identification String

You should set the server identification (ID) string to indicate the physical location of the server. This might include a building name, room number, or communication rack. This facilitates locating servers in a large network. By using the ID string, you don't have to worry about naming the server to indicate geographic location. If you later move a server, you can simply change its ID string. The sample syntax for an ID string is as follows:

```
> CHANGE SERVER ID "4TH FLOOR COMM. ROOM"
```

Circuit Timer

When a terminal server transmits information to a LAT host, the server encapsulates the data into an Ethernet packet. An Ethernet LAT packet contains approximately 25 bytes of addressing information and other overhead, leaving a minimum of 39 bytes available for user data. Regardless of how much data a terminal server has to send, it must send at least 64 bytes to meet the design requirements of Ethernet. If necessary, LAT adds null characters (padding) to a packet to meet this requirement. If a terminal server transmitted a packet *immediately* after a user typed a character, it would have to send 63 bytes of overhead just to send 1 byte of data. To better utilize the available space in a packet, terminal servers buffer characters from all ports for 80 msec before sending a packet. On a terminal server with many active ports to a common host, each transmitted packet usually contains data buffered from many ports.

Although servers buffer characters for 80 msec by default, you can adjust the buffer by changing the value of the server *circuit timer* parameter. A larger value forces a longer delay and usually reduces network traffic. However, if

you increase the buffer too much, users may begin to notice a delay between when they enter a character and when it is displayed on their terminals. In an OpenVMS environment, character echoing is the responsibility of the host. Every character that a user types is first sent to the host, the host optionally echoes the character back to the terminal, and the terminal prints it on the screen. This requires some overhead, but it allows the host to suppress character echo when appropriate (e.g., when entering passwords). A sample syntax is as follows:

```
> CHANGE SERVER CIRCUIT TIMER 100
```

Review Retransmit Limit Parameter

Many Digital networks have been extended to bridge LAT traffic over relatively low-speed, wide-area network (WAN) connections. Because the speed of these links is usually far less than that of Ethernet (56 Kb/sec or 1.5 Mb/sec versus 10 Mb/sec), these links are more susceptible to overutilization. When LAT traffic attempts to traverse an overutilized WAN link, packets are frequently dropped.

When a packet is transmitted by a terminal server, it should be acknowledged within 50–100 msec. If the original packet never reaches its destination or the returning acknowledgment is lost, the server retransmits the packet. The terminal server continues to retransmit with a 1-second delay between retransmissions up to eight times (or the setting of the retransmit limit). If the retransmit limit is exceeded, the terminal server assumes that the connection to the host is no longer reliable and terminates the virtual circuit to that host. This results in termination of *all* sessions from that terminal server to the host!

If someone performs a large file transfer over a network link that is near its capacity, all LAT sessions using that same link may time out. The default retransmit limit of 8 allows LAT connections to withstand network anomalies of approximately 8 seconds (8 * 1 second between retransmits). If you frequently experience LAT time-outs you may want to consider increasing retransmit limit to 30 or even 100 in extreme cases. If you use automatic LAT fail-over, a terminal server performs its retransmission cycle *before* it tries to fail-over to another host. Thus, if you set the retransmit limit too high, fail-over can take several minutes.

A sample syntax is as follows:

```
> CHANGE SERVER RETRANSMIT LIMIT 50
```

Configure LAT Groups

In a large LAT network you probably need to use LAT group codes to partition the network. The specific procedures for using LAT groups are discussed later in this chapter.

19.5.4 Configuring Terminal Server Ports

Usually, the default port settings allow interactive terminal connections. Use the SHOW PORT command to view a port configuration, as follows:

```
WN5104 1>> SHOW PORT 1

Port 1:                    B       Name:                  PORT_01

Character Size:            8       Speed:                    9600
Parity:                 None       Flow Control:              XON
Stop Bits:           Dynamic       Lines x Columns:       24 x 80

Access:                 Local      Type:                  Dynamic
Break Key:              Local      Terminal:                 ansi
Backwards Switch:        None      Session Limit:               4
Forwards Switch:         None      Prompt:            "%N %p>%P "
Local Switch:            None      Default Protocol:          ANY

Dedicated Service:       None
Preferred Service:       None

Authorized Groups:      10-20
Current Groups:         10-20

Enabled Characteristics:
  Autoprompt, Broadcast, Failover, Lock, Loss Notification,
  Message Codes, Command Completion, Verification, Input Flow Control,
  Output Flow Control
```

Of the foregoing parameters, the following are the key settings to allow interactive terminal access:

```
SPEED:           9600    (Set to match terminal speed or Enable AUTOBAUD)
CHARACTER SIZE:  8       (Set to match terminal)
PARITY:          NONE    (Set to match terminal)
ACCESS:          LOCAL   (Specifies access from the server port)
```

In configuring a port for a printer connection, the key difference is that ACCESS must be set to REMOTE. This allows connections from the network

(a LAT host) to the port. The procedures for configuring LAT printer ports are discussed in chapter 8.

19.5.5 Managing Service Nodes (Hosts)

In a LAT environment, terminal servers are usually considered the master, and hosts are considered slaves. Master nodes dictate the settings to be used for a LAT connection. Thus, settings for many LAT parameters (e.g., circuit timer, retransmit limit) are specified by the terminal server, leaving few parameters to be configured on the service node.

The following sections address the procedures for configuring the few LAT parameters that must be set on service nodes.

19.5.5.1 Setting LAT Node Name

Each LAT service node can have only one node name. By default, LAT defines its node name during LAT startup based on the setting of the system parameter SCSNODE (the VMScluster node name for the system). You can optionally select a different node name by passing it as a parameter to the LAT startup procedure as follows:

```
$ @SYS$STARTUP:LAT$STARTUP SALES BIRCH
```

The foregoing command starts LAT on the node. The service node name is BIRCH, and it offers the service SALES.

19.5.5.2 Assigning Service Names

You can configure a LAT node to offer one or more service names. By default, LAT hosts offer a service name that is the same as the node name. For example, node BIRCH offers the service name BIRCH.

If you run multiple applications on a single system or VMScluster, you may want to consider creating a service name for each application. For example, consider a cluster consisting of two nodes BIRCH and LOTUS that provide word processing, payroll, and sales forecasting applications. You could configure the cluster to offer three different LAT services (WP, PAYROLL, and SALES). You would then tell the respective users of each application to issue CONNECT commands to the appropriate service name. At some future point, you could move these three applications to three physically different systems (e.g., BAMBOO, HEMLCK, and MAPLE). You would then disable the WP, PAYROLL, and SALES services on BIRCH and LOTUS and enable them on the new

systems. Because users would still connect to the same service names, the change would be transparent to them.

19.5.5.3 Assigning Service Names on VMSclusters

If you have two or more nodes in a common-environment VMScluster you probably want to have all cluster members broadcast the same service name. For example, both nodes in a two-node cluster (BIRCH and LOTUS) offer the same service name (e.g., SALES). This allows users to issue connect requests to SALES and not be concerned with to which physical node they are connected.

Using this kind of service naming strategy greatly simplifies future hardware upgrades or changes. You can easily add a third cluster member, and have it offer the same SALES service. You effectively increase capacity without users' knowing it.

As the load on a service node increases, the LAT ratings of the services it offers decrease. In the following example, many users have logged into BIRCH, resulting in a drop in its LAT rating from 83 to 51.

```
TS001> SHOW SERVICE ALL STATUS

Service SALES - Available

Node Name       Status          Rating    Identification

BIRCH           Reachable       51        Welcome to VAX/VMS V6.1
LOTUS           Reachable       83        Welcome to VAX/VMS V5.5-2
```

Because LOTUS is now more available than BIRCH, any future connections to the SALES service result in a connection to LOTUS, as follows:

```
TS001> CONNECT SALES

Local -051- Session 1 [SALES] established

      Welcome to VAX/VMS V5.5-2 (LOTUS)

Username:
```

19.5.5.4 Starting LAT

Although DECnet and LAT are independent protocols, they usually share the same Ethernet adapter. Because DECnet modifies the datalink address of the controller when it starts, you must start DECnet before any other protocol.

Assuming DECnet is already started, you can start LAT from your system startup procedures, as follows:

```
$ @SYS$STARTUP:LAT$STARTUP     ! Site-independent LAT Startup
```

The LAT startup procedure executes the following two procedures:

```
SYS$STARTUP:LAT$CONFIG         ! Initialize LAT drivers
SYS$STARTUP:LAT$SYSTARTUP      ! Site-specific procedures
```

Because of the similarities in name, system managers sometimes inadvertently try to start LAT by executing LAT$SYSTARTUP instead of LAT$STARTUP. When this happens, you get an error message because LAT$CONFIG has not been executed. Make sure you start LAT using LAT$STARTUP.

If you have problems starting DECnet, check to see if LAT was somehow started first, as follows:

```
$ ANALYZE/SYSTEM
SDA> SHOW LAN
LAN Data Structures
-------------------
LSB address = 8084A300
Device state = 0013 Inited,Run,Timer

            -- ESA Unit Summary 14-APR-1994 13:17:40 --

UCB   UCB Addr   Fmt   Value    Client    State
-------------------------------------------------------------------
ESA0  80825B80
ESA2  807DF440   Eth   60-04    LAT       0015 Strtn,Uniq,Strtd
ESA3  8082F880   Eth   60-03    DECNET    0017 Strtn,Len,Uniq,Strtd
```

Look for the Ethernet devices (e.g., ESA2, ESA3). If LAT is listed first, you attempted to start LAT before DECnet. To resolve the problem, modify your system startup procedures to start DECnet first and reboot the system. The SHOW LAN command is undocumented and unsupported.

19.5.5.5 Stopping LAT

Stop LAT using the following command:

```
$ RUN SYS$SYSTEM:LATCP
LATCP> SET NODE/STATE=OFF
```

CAUTION: If you stop LAT on an active node, all LAT sessions to that host are terminated, and all LAT print queues stop. If all your users access the system from LAT terminal servers, stopping LAT has the same effect as shutting down the system. Don't stop LAT if you are logged in to the same host using LAT—you will terminate your own session and you won't be able to log back in. If you have to restart LAT and your only connection to that host is via LAT, submit your commands to stop and restart LAT as a batch job.

You can tell if any users have LAT connections by using the DCL SHOW USERS/FULL command to look for LTAnn: devices. However, even if there are no user connections, be aware that LAT print queues do not normally restart automatically when LAT is stopped and restarted. If you use normal (non-AUTOSTART) print queues you have to manually issue a START/QUEUE for every print queue on your system. If you have a lot of queues, you might be better off rebooting the entire system.

[OpenVMS 5.5]

If you use AUTOSTART print queues, you can restart all LAT print queues by issuing the ENABLE AUTOSTART/QUEUES DCL command (see chapter 8 for more information on print queues).

19.5.6 Configuring LAT Groups

In a large LAT network you can easily have hundreds of services offered by a variety of nodes. Some servers (e.g., DECserver 100s) have limited memory and run out of memory as the number of services increases. A large number of services can also be confusing to users. You might want to simplify their view of the network to include only the LAT services to which they need access.

LAT group codes address both of these issues by assigning group codes to service nodes, servers, and server ports. Each of these entities can be assigned one or more group codes consisting of the numbers 0–255. LAT groups are assigned at three layers, as follows:

1. Service Node Groups (set on a service node). By default, service nodes are enabled for group 0 only. When a node sends a LAT service announcement, it sets a bit to indicate each group for which it is enabled. You control service node groups by modifying the contents of SYS$STARTUP: LAT$SYSTARTUP.COM. For example, the following command enables the service node for groups 10 through 15:

Table 19.4 Restricting Access with LAT Groups

From Server/Port	*Access to BIRCH?*	*Access to LOTUS?*
TS001	Yes	Yes
TS002	Yes	No
TS001/PORT_1	Yes	Yes
TS001/PORT_2	Yes	No
TS001/PORT_3	No	Yes
TS002/PORT_3	No	No

```
$lcp set node /connections=incoming /groups=enable=(10-15)
```

2. Server Groups (set on a terminal server). By default, server groups are enabled for group 0 only. When a terminal server receives a LAT service announcement, the server compares the group code bits in the packet to its enabled groups. If there is at least one match, the server adds the service (defined in the announcement) to its tables. If the no group codes match, the server ignores the announcement. In this way terminal servers effectively filter out all services for which they are not group members. You control terminal server groups by entering the following commands on each terminal server:

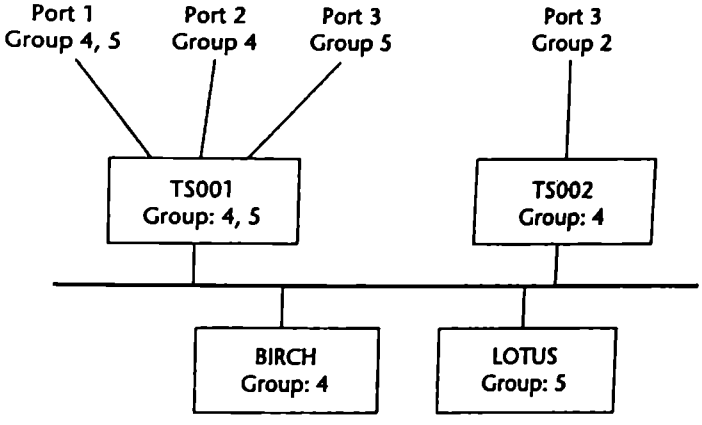
```
TS001> CHANGE SERVER GROUPS 10-15
```

3. Port Groups (set on terminal server ports). Service node groups and server groups control which services are added to the tables of a terminal server. Port groups determine which subset of those services individual users attached to each port can access. To access a service known to a server, a port must share at least one group with the service node group. You control port groups by entering the following commands from a terminal server:

```
TS001> CHANGE PORT # AUTHORIZED GROUPS 10-11
       (where #= port number)
```

Table 19.4 shows the access that would be allowed by the group configuration indicated in Figure 19.7.

Figure 19.7 Example LAT Groups



19.5.7 Troubleshooting LAT

The following are some typical LAT error messages and techniques for resolving them:

```
(1) Local -100- Access to service XXXXX denied
```

The server knows about the service, but port groups deny access. Compare the enabled port groups with the service node groups.

```
(2) Local -030- Unknown service/host "XXXXXX"
```

Server groups may prevent server from receiving service announcements. Compare server groups with service node groups.

Service announcements may not be reaching server. Do terminal servers on the same segment as the service node know about the service? If so, then the service node is operating properly and service announcements are probably being dropped by an intervening device (e.g., a bridge).

```
(3) Local -012- Session 1 [XXXX] disconnected
```

Session timeouts are a common problem in extended networks that bridge over low-speed links (e.g., 56 Kb/sec). LAT works fine at these speeds, as long as the links are not overutilized. You can decrease the sensitivity of LAT by increasing the retransmit limit parameter on every terminal server in your

network, as discussed earlier in this chapter. However, this only masks a larger problem of link overutilization. You should regularly measure bridge link utilization and error counts (e.g., number of dropped packets). Remember, LAT session can time-out in less than 10–15 *seconds*. The fact that a bridge might indicate that average utilization was only 10% over the last 10 *minutes* is meaningless. If you see a large number of dropped packets with low average utilization, you are probably experiencing short bursts of heavy traffic.

19.6 Additional References

Chappell, Laura, *Novell's Guide to Netware LAN Analysis*, Novell, 1993.
LAT Network Concepts, Digital, June 1987, order #AA-HY66A-TK.
Stevens, W. Richard, *TCP/IP Illustrated*, vol. 1, Addison-Wesley, 1994.

[OpenVMS 5.0]

Guide to DECnet-VAX Networking, Digital, April 1988.
Networking Manual, Digital, April 1988.

[OpenVMS 6.0]

DECnet for OpenVMS Guide to Networking, Digital, March 1994.
DECnet for OpenVMS Network Management Utilities, Digital, March 1994.
DECnet for OpenVMS Networking Manual, Digital, March 1994.
OpenVMS System Manager's Manual: Tuning, Monitoring, and Complex Systems, chapters 21–22, Digital, March 1944.

Appendix A
CONTACT INFORMATION

The following is a list of vendors referenced throughout this book.

Cahners Publishing Company
275 Washington St.
Newton, MA 02158
(617) 964-3030

Product: *Digital News & Review* (magazine)

Cardinal Business Media, Inc.
1300 Virginia Dr., Ste. 400
Ft. Washington, PA 19034
(215) 643-8000

Product: *DEC Professional* and *Digital Systems Journal* (magazines)

Data Center Software
70 Herrick St.
Beverly, MA 01915
(800) 232-8297
(508) 922-5500

Product: QueMAN

DECUS
Digital Equipment Corporation Users Society
333 South St. SHR1-4
Shrewsbury, MA 01545
(800) DECUS55
(508) 841-3570

Products: User group, public domain software, symposia

Digital
Corporate Customer Relations
(800) DEC-INFO
(800) 332-4636

Description: Corporate Customer Relations is a last resort if you are unable to solve a problem or get information from other Digital sources.

Digital
DECdirect
(800) 344-4825
(800) 234-1998 (modem)

The following catalogs are available at no cost:

DECdirect Hardware Edition Catalog
DECdirect Software Edition Catalog
DECdirect Workstation Edition Catalog
DECdirect RISC Workstations and Applications Catalog
DECdirect Personal Computing Integration Software and Hardware Catalog
Digital Assisted Services U.S. Edition Price List
Software Documentation Products Catalog

Digital
Printing and Circulation
(508) 351-4401

The following information is available from DEC Printing at no cost:

Integrated Promotional Literature Catalog (May 1989)
VAX/VMS Software Source book
VAX Systems/DECsystems Systems and Options Catalog
Telecommunications and Networks Buyer's Guide

Target Systems Corporation
33 Boston Post Road West
Marlboro, MA 01752
(800) 233-3493
(508) 460-9206

Product: Target-Batch

Reliability Ratings, Inc.
163 Highland Ave.
Needham, MA 02194
(617) 444-5755

Product: Equipment reliability statistics

Appendix B
SITE GUIDE OUTLINE

The following is an outline for a site guide, as discussed in chapter 14.

1.0 Site and Hardware Configuration

Equipment inventory
Equipment layout diagrams
Electrical system
Uninterruptible power supply (UPS), conditioning,
emergency shutoff, alarms
Air conditioning—BTU requirements, current usage, alarms
Safety equipment
Halon, fire extinguishers, heat, smoke, water alarms

2.0 VMS Configuration

Versions in use on each system
Special tailoring
Patches applied
System parameter settings
Page/swap file sizing

3.0 System Boot Procedures

Console subsystem initialization
Configuration of front panel switches
Normal boot procedures
Conversational boot procedures
Stand-alone backup boot procedures

4.0 System Start-up Procedures

Flow chart of start-up procedures
Special order dependencies

5.0 System Shutdown Procedures

Pre-shutdown procedures
Configuration of SYSHUTDWN.COM
Normal shutdown
Shutdown with reboot

6.0 Installed Optional Software Products

Details on each installed optional product
Date, version, location, start-up, shutdown procedures
Required VMS version, SYSGEN, and user authorization file (UAF) requirements

7.0 User Account Management Procedures

Username formats
Single ownership of accounts
Pre-expiration of new passwords
Organization of user identification code (UIC) groups
Configuration of the DEFAULT account
Password policy

8.0 System Login Procedures

Flow chart of login process
Global symbols defined
Assigning default print queue

9.0 Departmental Administrative Issues

Staff organization chart
Outline of staff responsibilities
Maintenance agreement strategies

10.0 Operation Production Jobs

Production job schedule
Servicing job requests

Job rerun procedures
Handling job exceptions

11.0 Backups

Requirements dictated by business
Backup strategies
Daily backup procedures

12.0 Guide to Support Resources

Internal
DEC and third party
Documentation inventory
Books and publications

13.0 System Integrity Monitoring

Procedures for detecting and handling the following:
- Disk errors
- Low disk space
- Failed job or process
- Runaway jobs
- Network failures

14.0 Security Procedures

Requirements of business
Security strategy
Security procedures
Default file protection
Specific protection requirements for major areas

15.0 Performance Management

Tuning and capacity history
Known bottlenecks
Current capacity and estimated future growth

Appendix C
SYSTEM CONSOLE SUMMARY

Most system management tasks are similar whether you manage a MicroVAX or VAX 6000. However, boot procedures can vary dramatically from machine to machine because of the variety of system buses to which a boot device might be attached. Fortunately, systems with the same bus type have similar boot procedures. As a result, boot procedures described in chapter 4 are organized by the various system bus types described in Table C.1.

Table C.1 VMS System Bus Types

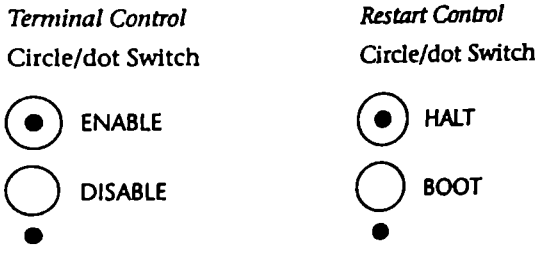
Bus	*Systems*
QBUS	MicroVAX 2000, VAXstation II/GPX, Microvax 3500/3600, VAXstation 3200/3500, MicroVAX 3800/3900
SCSI	MicroVAX 3100, VAXstation 3100, VAXstation 4000, VAX 4000-100, VAXstation 3520/3540
DSSI	MicroVAX 3300/3400, VAX 4000, VAXft 3000
UNIBUS	VAX 11/730, VAX 11/780, VAX 11/780, VAX 11/785, VAX 8600/8650
BI	VAX 8200/8300, VAX 8250/8350, VAX 8500/8530/8550, VAX 8700/8800, VAX 8810/8820/8830/8840
XMI	VAX 6000, VAX 9000
ALPHA	AXP 3000, AXP 4000, AXP 7000, AXP 10000

C.1 QBUS-based Systems

MicroVAX II/VAXstation II

Console CPU	Main CPU	
Console Operating System	N/A	
Console Modes	Console or Program	
Console Load Device	ROM	
Terminal Control	Circle/dot Switch	
System Restart Control	Circle/dot Switch	
Console Mode Switch	<BREAK> *or* HALT twice	
Program Mode Switch	`>>> CONT`	
Default Boot Device	N/A	
Default Boot Control Flags	N/A	
Power-on Boot	Sniffer Boot	
Manual Boot Command	`>>> B`	Same as power-on Boot
	`>>> B device`	Boot from device
	`>>> B/xxxxxxxx`	Manually specify control flags
Valid Devices and Sniffer Boot Order	`DUA2, DUA0, DUA1, MUA0, ESA0` (Ethernet)	

Terminal and Restart Control



Sniffer Boot

On these systems you cannot configure a default boot device. Rather, the system attempts to boot from available devices in the order indicated in the above list.

MicroVAX 2000/VAXstation 2000

Console CPU	Main CPU
Console O/S	N/A
Console Modes	Console or Program
Console Load Device	ROM
Terminal Control	None
System Restart Control	`>>> TEST 53`
Console Mode Switch	Press HALT once

Program Mode Switch	`>>> CONT` .	
Default Boot Device	`>>> TEST 51`	
Default Boot Control Flags	`>>> TEST 52`	
Power-on Boot	Boot from Default Boot Device	
Manual Boot Command	`>>> B`	Sniffer Boot (Note: *not* same as Power-on)
	`>>> B device`	Boot from device
	`>>> B/flags`	Manually specify Control Flags
Valid Devices and Sniffer Boot Order	`DUA0, ..., DUB0, ..., MUA0, XQA0` (Ethernet)	

Sniffer Boot

On these systems a sniffer boot is initiated when you issue a manual boot (e.g., >>> B)—even if you have configured a default boot device. A sniffer boot attempts to successively boot from all available devices in the order indicated in the foregoing above. If you need to perform a manual boot, be sure to explicitly specify a device; otherwise the system performs a sniffer boot and possibly boots from a device other than the one you intended.

Configuring Default Boot Device

`>>> TEST 51`	Displays current default boot device
`DUA0?>>> device`	Enter new boot device name to change or press <RETURN> to retain current setting

Configuring Default Boot Control Flags

`>>> TEST 52`	Displays current setting for default boot control flags
`00000000?>>> control flags`	Enter new control flags or press <RETURN> to retain current setting

Configuring System Restart Control

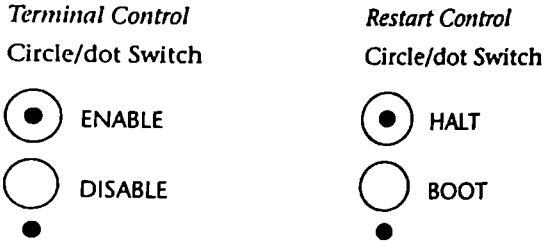
`>>> TEST 53`	Displays current setting for restart control
`2?>>> restart value`	Enter new restart value (2=BOOT, 3=HALT) *or* press <RETURN> to retain current setting

MicroVAX 3500/3600, VAXstation 3200/3500

Console CPU	Main CPU
Console O/S	N/A
Console Modes	Console or Program
Console Load Device	ROM
Terminal Control	Circle/dot Switch
System Restart Control	Circle/dot Switch
Console Mode Switch	<BREAK> *or* HALT twice
Program Mode Switch	`>>> CONT`
Default Boot Device	`>>> SET BOOT device`

Default Boot Control Flags	`>>> SET BFLG control-flags`	
Power-on Boot	Boot from Default Boot Device	
Manual Boot Command	`>>> B`	Sniffer boot
	`>>> B device`	Boot from device
	`>>> B/xxxxxxxx`	Manually specify control flags
Valid Devices and Sniffer Boot Order	`Device Unit Numbers 0 .. 15`	

Terminal and Restart Control



Sniffer Boot

On these systems a sniffer boot is initiated when you issue a manual boot (e.g., >>> B)—even if you have configured a default boot device. A sniffer boot attempts to successively boot from all valid devices in the order indicated in the table. If you need to perform a manual boot, be sure to explicitly specify a device, otherwise the system performs a sniffer boot and possibly boots from a device other than the one you intended.

Configuring Default Boot Device

`>>> SHOW BOOT`	Displays current default boot device
`>>> SHOW DEVICE`	Displays connected devices
`>>> SET BOOT device`	Sets default boot device

Configuring Default Boot Control Flags

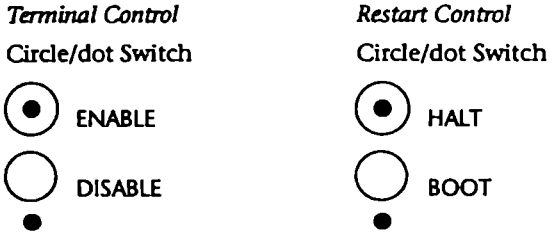
`>>> SHOW BFLG`	Displays current setting for default boot control flags
`>>> SET BFLG control-flags`	Sets default boot control flags

MicroVAX 3800/3900

Console CPU	Main CPU
Console O/S	N/A
Console Modes	Console or Program
Console Load Device	ROM
Terminal Control	Circle/dot Switch
System Restart Control	Circle/dot Switch

Console Mode Switch	<BREAK> *or* HALT twice
Program Mode Switch	`>>> CONT`
Default Boot Device	`>>> SET BOOT device`
Default Boot Control Flags	`>>> SET BFLG control-flags`
Power-on Boot	Boot from Default Boot Device, if configured, otherwise Query Boot
Manual Boot Command	`>>> B` Same as Power-on Boot `>>> B device` Boot from device `>>> B/xxxxxxxx` Manually specify Control Flags
Valid Devices and Sniffer Boot Order	`DJcu, DUcu, MUcu, XQAu` (where c = Controller and u = Unit)

Terminal and Restart Control



Query Boot

When powered on, the system boots from the default boot device. If the device is not configured, the system displays available devices on the console terminal and waits 30 seconds for input. If no input is received within the allowed time, the system attempts to boot from the network device.

Configuring Default Boot Device

`>>> SHOW BOOT`	Displays current default boot device
`>>> SHOW DEVICE`	Displays connected devices
`>>> SET BOOT device`	Sets default boot device

Configuring Default Boot Control Flags

`>>> SHOW BFLG`	Displays current setting for default boot control flags
`>>> SET BFLG control-flags`	Sets default boot control flags

C.2 SCSI-based Systems

MicroVAX/VAXstation 3100-30,38,48

Console CPU	Main CPU
Console O/S	N/A
Console Modes	Console or Program
Console Load Device	ROM
Terminal Control	N/A
System Restart Control	`>>> SET HALT restart-value`
Console Mode Switch	<BREAK> *or* HALT button
Program Mode Switch	`>>> CONT`
Default Boot Device	`>>> SET BOOT device`
Default Boot Control Flags	`>>> SET BFLG control-flags`
Power-on Boot	Boot from default boot device *or*, if default device is not configured, network device
Manual Boot Command	`>>> B` Same as power-on boot `>>> B device` Boot from device `>>> B/xxxxxxxx` Manually specify control flags
Valid Devices	`DKAu00, DKBu00, ESA0, MKAu00, MKBu00` (where u = Unit)

Configuring Restart Control

`>>> SHOW HALT`	Displays current setting for restart control
`>>> SET BFLG control-flags`	Sets default boot control flags (where 2 = BOOT, and 3 = HALT)

Configuring Default Boot Device

`>>> SHOW BOOT`	Displays current default boot device
`>>> SHOW DEVICE`	Displays connected devices
`>>> SET BOOT device`	Sets default boot device

Configuring Default Boot Control Flags

`>>> SHOW BFLG`	Displays current setting for default boot control flags
`>>> SET BFLG control-flags`	Sets default boot control flags

MicroVAX/VAXstation 3100-40, 76, 80, 4000-60, 4000-VLC

Console CPU	Main CPU
Console O/S	N/A
Console Modes	Console or Program
Console Load Device	ROM
Terminal Control	N/A
System Restart Control	`>>> SET HALT restart-value`
Console Mode Switch	<BREAK> *or* HALT button
Program Mode Switch	`>>> CONT`

Default Boot Device	`>>> SET BOOT device1, device2`	
Default Boot Control Flags	`>>> SET BFLG control-flags`	
Power-on Boot	Boot from default boot device *or*, if default device is not configured, the network device	
Manual Boot Command	`>>> B`	Same as power-on boot
	`>>> B device`	Boot from device
	`>>> B/xxxxxxxx`	Manually specify control flags
Valid Boot Devices	`DKAu00 or MKAu00` (where u = Unit, can boot only from controller A)	

VAXstation 3520/3540

Console CPU	Main CPU	
Console O/S	N/A	
Console Modes	Console or Program	
Console Load Device	ROM	
Terminal Control	N/A	
System Restart Control	`CPUxx>>> SET HALT restart value`	
Console Mode Switch	<BREAK> *or* HALT button	
Program Mode Switch	`CPUxx>>> CONT`	
Default Boot Device	`CPUxx>>> SET BOOT device1, device2`	
Default Boot Control Flags	`CPUxx>>> SET BFLG control-flags`	
Power-on Boot	Boot from default boot device *or*, if default device is not configured, query boot	
Manual Boot Command	`CPUxx>>> B`	Not valid (must specify device)
	`CPUxx>>> B device`	Boot from device
	`CPUxx>>> B/xxxxxxxx`	Manually specify control flags
Valid Boot Devices	`DKA000, DKA100 - DKA700, ESA0, MUA0`	

C.3 DSSI-based Systems

MicroVAX 3300/3400

Console CPU	Main CPU
Console O/S	N/A
Console Modes	Console or Program
Console Load Device	ROM
Terminal Control	N/A
System Restart Control	`>>> SET HALT restart value`
Console Mode Switch	<BREAK> *or* HALT button
Program Mode Switch	`>>> CONT`
Default Boot Device	`>>> SET BOOT device`
Default Boot Control Flags	`>>> SET BFLG control-flags`
Power-on Boot	Boot from default boot device *or*, if default device is not configured, boot from network
Manual Boot Command	`>>> B` Not valid (must specify device) `>>> B device` Boot from device `>>> B/xxxxxxxx` Manually specify control flags
Valid Boot Devices	`[node$]DIAu, DUcu, DLcu, [node$]MIAu, MUcu, ESA0, XQcu`

VAX 4000 Model 200,300,400,500,600

Console CPU	Main CPU
Console O/S	N/A
Console Modes	Console or Program
Console Load Device	ROM
Terminal Control	N/A
System Restart Control	`>>> SET HALT restart value`
Console Mode Switch	<BREAK> *or* HALT button
Program Mode Switch	`>>> CONT`
Default Boot Device	`>>> SET BOOT device1, device2`
Default Boot Control Flags	`>>> SET BFLG control-flags`
Power-on Boot	Boot from default boot device *or* if default device is not configured, perform query boot
Manual Boot Command	`>>> B` Not valid (must specify device) `>>> B device` Boot from device `>>> B/xxxxxxxx` Manually specify control flags
Valid Boot Devices	`[node$]DIcu, DUcu, DKAu, MIcu, MUcu, EZA0, XQcu`

VAXft 3000-310

Console CPU	Main CPU
Console O/S	N/A
Console Modes	Console *and* Program

Console Load Device	ROM
Terminal Control	Local console disable switch Pushed in: DISABLED Pulled out: ENABLED
System Restart Control	N/A
Console Mode Switch	<BREAK> key
Program Mode Switch	`>>> PIO or SET TERM PROG`
Default Boot Device	`>>> SET BOOT/P DEFAULT device`
Default Boot Control Flags	`>>> SET BOOT/P DEFAULT /R5:xxxxxxxx device`
Power-on Boot	Boots using boot parameters associated with the "DEFAULT" boot name
Manual Boot Command	`>>> B` Same as power-on boot `>>> B boot-name` Boot using defined boot-name
Valid Boot Devices	`DIcu, MIcu, EPcu` (where c = Controller (A,B,F, or G) and u = Unit)

C.4 XMI-based Systems

VAX 6000

Console CPU	Main CPU
Console O/S	N/A
Console Modes	Console *or* Program
Console Load Device	ROM or TK tape drive
Terminal Control	*Upper Front Panel Switch* — *Terminal Control* ENABLE — ENABLED SECURE — DISABLED
System Restart Control	If terminal control DISABLED, then BOOT, otherwise depends on lower switch settings: *Lower Front Panel Switch* — *Boot Action* UPDATE — HALT HALT — HALT AUTO START — BOOT
Console Mode Switch	CTRL/P *or* HALT button
Program Mode Switch	`>>> CONT`
Default Boot Device	`>>> SET BOOT DEFAULT /... device`
Default Boot Control Flags	`>>> SET BOOT DEFAULT /R5:xxxxxxxx/... device`
Power-on Boot	Boots using boot parameters associated with the "DEFAULT" boot name
Manual Boot Command	`>>> B` Same as power-on boot `>>> B boot-name` Boot using defined boot-name
Valid Boot Devices	`DUu, ET0, MU0, CSu` (where u = Unit number)

Default Boot Device and Control Flags

You can display the currently defined boot names using the following command:

```
>>> SHOW BOOT

DEFAULT    /R5:00000000 /XMI:E /BI:4 /NODE:00000100 DU0
STAB       /R5:E0000000 /XMI:E /BI:4 /NODE:00000100 DU0
CONV       /R5:00000001 /XMI:E /BI:4 /NODE:00000100 DU0
```

By default, the system boots using the parameters and device specified by the DEFAULT boot name. You can modify the default parameters using the following command:

```
>>> SET BOOT DEFAULT/XMI:x/BI:y/NODE:z/R3:a/R5:b DUu

(Where     u = Disk unit #
           x = XMI adapter #
           y = BI Bus #
```

```
n = HSC node list
a = Value to pass to R3 (for Phase I shadowing)
b = Value to pass to R5 (boot control flags)
```

Creating Additional Boot Names

The terminal control switch must be set to the "Update" position before you can modify boot names.

```
>>> SET BOOT CONV/XMI:E/BI:4/NODE:00000100/R5:00000001 DU0
>>> BOOT CONV
>>> SET BOOT STAB/XMI:E/BI:4/NODE:00000100/R5:E0000000 DU0
>>> BOOT STAB
>>> SET BOOT ALT/XMI:E/BI:4/NODE:00000100/R5:00000000 DU1
>>> BOOT ALT
```

VAX 9000

Console CPU	Modified MicroVAX II	
Console O/S	VAXeln	
Console Modes	Console *and* Program	
Console Load Device	RD54	
Terminal Control	Terminal control switch	
System Restart Control	Front panel switch: BOOT RESTART/BOOT RESTART/HALT HALT	
Console Mode Switch	CTRL/P *or* HALT button	
Program Mode Switch	`>>> CONT or SET TERM/PROG`	
Default Boot Device	Maintained in Boot Files on console device	
Default Boot Control Flags	Maintained in Boot Files on console device	
Power-on Boot	Executes DEFBOO.CMD boot file located on console device	
Manual Boot Command	`>>> B`	Same as Power-on Boot
	`>>> B xyz`	Executes xyzBOO.CMD Boot File
Valid Boot Devices		

C.5 ALPHA and VAX 7000/10000 Systems

Alpha AXP 3000

Console CPU	Main CPU
Console O/S	N/A
Console Modes	Console *or* Program
Console Load Device	N/A
Terminal Control	N/A
System Restart Control	N/A
Console Mode Switch	CTRL/P *or* HALT button
Program Mode Switch	`>>> CONT`
Default Boot Device	`>>> SET BOOTDEF_DEV device`
Default Boot Control Flags	`>>> SET BOOT_OSFLAGS control-flags`
Power-on Boot	Boots from default boot device
Manual Boot Command	`>>> B` Same as Power-on Boot `>>> B -fl r,f` Manually specify control flags (where r = System Root and f = Boot Control Flags)
Valid Boot Devices	Issue `>>> SHOW DEVICE` command

Default Boot Device and Control Flags

Configure these parameters by establishing values for the environment variables BOOTDEF_DEV and BOOT_OSFLAGS. You can view the current contents of all environmental variables by using the following command:

```
>>> SHOW
```

You can view a list of available devices by entering the following command:

```
>>> SHOW DEVICE
```

Finally, use the SET command to change environment variables, as follows:

```
>>> SET BOOTDEF_DEV DKA100
>>> SET BOOT_OSFLAGS A,1       ! Conversational boot from [SYSA]
```

Alpha AXP 4000

Console CPU	Main CPU
Console O/S	N/A
Console Modes	Console *or* Program

Console Load Device	N/A	
Terminal Control	N/A	
System Restart Control	N/A	
Console Mode Switch	CTRL/P *or* HALT button	
Program Mode Switch	`>>> CONT`	
Default Boot Device	`>>> SET BOOTDEF_DEV device`	
Default Boot Control Flags	`>>> SET BOOT_OSFLAGS control-flags`	
Power-on Boot	Boots from default boot device	
Manual Boot Command	`>>> B`	Same as Power-on Boot
	`>>> B -fl r,f`	Manually specify control flags (where r = System Root and f = Boot Control Flags)
Valid Boot Devices	Issue `>>> SHOW DEVICE` command	

Alpha AXP 7000, 10000 and VAX 7000, 10000

Console CPU	Main CPU	
Console O/S	N/A	
Console Modes	Console *or* Program	
Console Load Device	N/A	
Terminal Control	N/A	
System Restart Control	N/A	
Console Mode Switch	CTRL/P *or* HALT button	
Program Mode Switch	`>>> CONT`	
Default Boot Device	`>>> SET BOOTDEF_DEV device`	
Default Boot Control Flags	`>>> SET BOOT_OSFLAGS control-flags`	
Power-on Boot	Boots from default boot device	
Manual Boot Command	`>>> B`	Same as Power-on Boot
	`>>> B -fl r,f`	Manually specify control flags (where r = System Root and f = Boot Control Flags)
Valid Boot Devices	Issue `>>> SHOW DEVICE` command	

C.6 UNIBUS-based Systems

WAX 11/780

Console CPU	Dedicated PDP-11
Console O/S	N/A
Console Modes	Console *and* Program
Console Load Device	RX01 Floppy
Terminal Control	*Switch Position* — *Terminal Control*
	OFF — No power applied to CPU
	LOCAL — ENABLED
	LOCAL DISABLE — DISABLED
	REMOTE — ENABLED
	REMOTE DISABLE — DISABLED
System Restart Control	ON Equivalent to RESTART/BOOT
	OFF Equivalent to RESTART/HALT
Console Mode Switch	CTRL/P *or* HALT button
Program Mode Switch	`>>> SET TERM PROG`
Default Boot Device	Configured in boot file: DEFBOO.CMD
Default Boot Control Flags	Configured in boot file: DEFBOO.CMD
Power-on Boot	Executes boot file: DEFBOO.CMD
Manual Boot Command	`>>> B` — Same as Power-on Boot
	`>>> B xyz` — Executes boot file: xyzBOO.CMD
	`>>> @file` — Executes boot file: file
	`>>> B CS1` — Boot Stand-alone Backup from RX01
Template Boot Files	
Massbus #1	`DB0BOO.CMD`
Massbus #2	`DBBB00.CMD`
Unibus #1	`DU0BOO.CMD (for RA-style disks)`
Unibus #2	N/A
CI	`CIBOO.CMD (for HSC-attached disks)`

VAX 11/785

Console CPU	Dedicated PDP-11
Console O/S	N/A
Console Modes	Console *and* Program
Console Load Device	RX01 Floppy
Terminal Control	*Switch Position* — *Terminal Control*
	OFF — No power applied to CPU
	LOCAL — ENABLED
	LOCAL DISABLE — DISABLED
	REMOTE — ENABLED
	REMOTE DISABLE — DISABLED

System Restart Control	ON	Equivalent to RESTART/BOOT
	OFF	Equivalent to RESTART/HALT
Console Mode Switch	CTRL/P *or* HALT button	
Program Mode Switch	`>>> SET TERM PROG`	
Default Boot Device	Configured in boot file: DEFBOO.CMD	
Default Boot Control Flags	Configured in boot file: DEFBOO.CMD	
Power-on Boot	Executes boot file: DEFBOO.CMD	
Manual Boot Command	`>>> B`	Same as Power-on Boot
	`>>> B xyz`	Executes boot file: xyzBOO.CMD
	`>>> @file`	Executes boot file: file
	`>>> B XYu`	Executes XYBOO.CMD and boot from drive unit "u"
	`>>> B CS1`	Boot Stand-alone Backup from RX01
Template Boot Files		
Massbus #1	`DBABOO.CMD`	
Massbus #2	`DBBB00.CMD`	
Unibus #1	`DUBOO.CMD (for RA-style disks)`	
Unibus #2	N/A	
CI	`CIBOO.CMD (for HSC-attached disks)`	

VAX 8600/8650

Console CPU	Dedicated PDP-11 (T-11) microcomputer	
Console O/S	RT-11	
Console Modes	Console *and* Program	
Console Load Device	RL02 removable disk	
Terminal Control	*Switch Position*	*Terminal Control*
	OFF	N/A
	LOCAL	ENABLED
	LOCAL/DISABLE	DISABLED
	REMOTE	ENABLED
	REMOTE/DISABLE	DISABLED
System Restart Control	*Switch Position*	*Restart Control*
	BOOT	BOOT
	RESTART/BOOT	RESTART/BOOT
	RESTART/HALT	RESTART/HALT
	HALT	HALT
Console Mode Switch	CTRL/P *or* HALT button	
Program Mode Switch	`>>> CONT`	
Default Boot Device	Configured in boot file: DEFBOO.COM	
Default Boot Control Flags	Configured in boot file: DEFBOO.COM	
Power-on Boot	Executes boot file: DEFBOO.COM	
Manual Boot Command	`>>> B`	Same as Power-on Boot
	`>>> B xyz`	Executes boot file: xyzBOO.CMD
	`>>> @file`	Executes boot file: file
	`>>> B CS1`	Boot Stand-alone Backup from RX01

Template Boot Files	
Massbus #1	DB0BOO.CMD
Massbus #2	N/A
Unibus #1	DU0BOO.CMD (for RA-style disks)
Unibus #2	N/A
CI	CIBOO.CMD (for HSC-attached disks)

C.7 BI-based Systems

VAX 8200/8300/8250/8350

Console CPU	Main CPU	
Console O/S	N/A	
Console Modes	Console *or* Program	
Console Load Device	2 RX50 floppy drives	
Terminal Control		
System Restart Control	Lower key: AUTO START - BOOT	
Console Mode Switch	CTRL/P *or* HALT button	
Program Mode Switch	`>>> CONT`	
Default Boot Device	Configured in boot file: DEFBOO.CMD	
Default Boot Control Flags	Configured in boot file: DEFBOO.CMD	
Power-on Boot	Executes boot file: DEFBOO.CMD	
Manual Boot Command	`>>> B`	Same as Power-on Boot
	`>>> B device`	Boot from local device
	`>>> B/r5:800 CSA1`	Invoke BOOT58
	`BOOT58> @file`	Executes specified boot file
	`>>> B CSA1`	Boot Stand-alone Backups from RX50
Template Boot Files		
BI Bus	`KDBBOO.CMD`	
CI Bus	`CIBOO.CMD`	

VAX 8500/8530/8550/8700/8800

Console CPU	Dedicated PDP-11 microcomputer (Pro 380)	
Console O/S	P/OS	
Console Modes	Console *and* Program	
Console Load Device	2 RX50 floppy drives and RD53	
Terminal Control	ENABLE: `>>> ENABLE LOCAL CONSOLE`	
	DISABLE: `>>> DISABLE LOCAL CONSOLE`	
System Restart Control	BOOT: `>>> ENABLE AUTO POWERON and`	
	`>>> ENABLE AUTO BOOT`	
	HALT: `>>> DISABLE AUTO BOOT`	
Console Mode Switch	CTRL/P	
Program Mode Switch	`>>> SET TERM PROG`	
Default Boot Device	Configured in boot file: DEFBOO.COM	
Default Boot Control Flags	Configured in boot file: DEFBOO.COM	
Power-on Boot	Executes boot file: DEFBOO.COM	
Manual Boot Command	`>>> B`	Same as Power-on Boot
	`>>> B XYZu`	Executes xyzBOO.COM and passes "u" as drive unit number (R3) XYZ can be: BDA, UDA, *or* BCI
	`>>> @file`	Executes specified boot file
	`>>> @CSBOO.COM`	Boot stand-alone backups from RX50

Template Boot Files	
BI Bus	`BDABOO.COM`
Unibus	`UDABOO.COM`
CI Bus	`BCIBOO.COM`

VAX 8810/8820/8830/8840

Console CPU	Dedicated MicroVAX II	
Console O/S	Modified VMS	
Console Modes	Console *and* Program	
Console Load Device	TK50 and RD53	
Terminal Control	ENABLE:	`>>> ENABLE LOCAL CONSOLE`
	DISABLE:	`>>> DISABLE LOCAL CONSOLE`
System Restart Control	BOOT:	`>>> ENABLE AUTO POWERON` and `>>> ENABLE AUTO BOOT`
	HALT:	`>>> DISABLE AUTO BOOT`
Console Mode Switch	CTRL/P *or* HALT button	
Program Mode Switch	`PS-CIO-0> CONT`	
Default Boot Device	Configured in boot file: `DEFBOO.COM`	
Default Boot Control Flags	Configured in boot file: `DEFBOO.COM`	
Power-on Boot	Executes boot file: `DEFBOO.COM`	
Manual Boot Command	`PS-CIO-0> B`	Same as Power-on Boot
	`PS-CIO-0> B XYZu`	Executes xyzBOO.COM and passes "u" as drive unit number (R3). XYZ can be: BDA, UDA, or BCI
	`PS-CIO-0> @file`	Executes specified boot file
	`PS-CIO-0> @CSABOO`	Boot stand-alone backup from RD53
Template Boot Files		
BI Bus	`BDABOO.COM`	
Unibus	`UDABOO.COM`	
CI Bus	`BCIBOO.COM`	

Appendix D
OpenVMS PRIVILEGES

OpenVMS privileges are separated into the following six categories:

```
NORMAL     ! Normal privileges
GROUP      ! Grants access to group resources
DEVOUR     ! Provides potential to consume system resources
SYSTEM     ! Potentially affects system processes and resources
FILE       ! Allows user to bypass file security
ALL        ! Allows user complete system access
```

Privileges were recategorized in OpenVMS 6.0, but the functions remain basically the same. The purpose of the following tables is to give you an idea of the DCL functions enabled by various privileges.

Table D.1 NORMAL privileges

Privilege	*Allowed Functions*	*Description*
NETMBX	$ SET HOST	Creates an interactive login to a remote system
TMPMBX	$ SPAWN	Creates a subprocess

Table D.2 GROUP privileges

Privilege	*Allowed Functions*	*Description*
GROUP	$ SHOW ENTRY $ STOP process	Affect objects in the same user identification code (UIC) group
GRPPRV	File operations	Access files in the same UIC group via SYSTEM access category

Table D.3 DEVOUR privileges

Privilege	*Allowed Functions*	*Description*
ACNT	$ RUN/DETACHED/NOACC	Bypass VMS Accounting
ALLSPOOL	$ ALLOCATE device	Allocate a spooled device
BUGCHK	N/A	Create bug check error log entries
EXQUOTA	$ CREATE	Exceed disk quota
GRPNAM	$ DEFINE/GROUP $ DISMOUNT/GROUP	Define group logical names
PRMCEB	N/A	Create permanent common event flag blocks
PRMGBL	$ INSTALL	Create and map global sections
PRMMBX	N/A	Create permanent mailboxes
SHMEM	N/A	Control multiport memory

Table D.4 FILE privileges

Privilege	*Allowed Functions*	*Description*
DIAGNOSE	$ SET HOST/DUP $ SET HOST/HSC	Run on-line diagnostic programs
VOLPRO	$ DISMOUNT/ABORT $ INITIALIZE	Override volume protection

Table D.5 SYSTEM privileges

Privilege	*Allowed Functions*	*Description*
ALTPRI	$ SET PROCESS/PRIORITY= $RUN/DETACHED/PRIORITY= $ SUBMIT/PRIORITY= $ SET ENTRY/PRIORITY	Select a current priority > then UAF base priority
OPER	$ DEFINE/CHARACTERISTIC $ DEFINE/FORM $ DELETE/CHARACTERISTIC $ DELETE/ENTRY $ DELETE/FORM $ DELETE/QUEUE $ INITIALIZE/QUEUE $ REPLY $ SET ACCOUNTING	Perform queue management and other operation functions

	$ SET CLUSTER/EXPECTED_VOTE	
	$ SET DAY	
	$ SET LOGINS	
	$ SET PRINTER	
	$ SET TIME	
	$ START/QUEUE	
	$ START/QUEUE/MANAGER	
	$ STOP/QUEUE	
PSWAPM	$RUN/DETACHED/NOSWAPPING	Disable process swapping
SHARE	$ OPEN	Access a device allocated by other processes
SYSLCK	N/A	Lock system-wide resources
WORLD	$ STOP process	Control processes outside your UIC

Table D.6 ALL privileges

Privilege	*Allowed Functions*	*Description*
BYPASS	Object access functions	Bypass UIC object protection
CMEXEC	N/A	Change mode to Executive
CMKRNL	$ DELETE/INTRUSION	Change mode to Kernel
	$ SET UIC	
	$ SET PASSWORD/SYSTEM	
	$ SHOW INTRUSION	
	$ START/CPU	
	$ STOP/CPU	
DETACH	$ RUN/DETACH/UIC=	Create a detached process under a different UIC
LOG_IO	$ SET PRINTER	Perform logical input-output (I/O) operations to a device
	$ SET TERM/HANGUP	
	$ SET TERM/PERM	
	$ SET TERM/NOSYSPASSWORD	
	$ SET TIME	
	$ SHOW TERM/PERM	
MOUNT	N/A	Issue a mount queued I/O (QIO)
PFNMAP	N/A	Map physical memory
PHY_IO	N/A	Perform physical I/O operations to a device
READALL	File operations	Allows READ and CONTROL access to all files
SECURITY	$ DELETE/INTRUSION	Perform security functions
	$ SET AUDIT	
	$ SET PASSWORD/SYSTEM	
	$ SHOW AUDIT	

	`$ SHOW INTRUSION`	
`SETPRV`	`$ SET PROCESS/PRIVILEGE=`	Enable any privilege
	`$ RUN/PRIVILEGES`	
`SYSNAM`	`$ DEFINE/SYSTEM`	Define system logical name
	`$ CREATE/NAME_TABLE/EXEC`	
	`$ DISMOUNT/SYSTEM`	
	`$ START/QUEUE/MANAGER`	
`SYSPRV`	`File operations`	Access files via SYSTEM access category

Appendix E
DISKETTE CONTENTS

The following sections describe the procedures for installing the sample command files located on the diskette included with this book. The 1.2 MB, 3.5-inch disk is MSDOS formatted and contains approximately 50 ASCII text files. Because of MSDOS file naming restrictions, the files on the disk are named using the format FILEnny.TXT (where *nn* represents the number of the chapter in which the example is discussed and *y* is a sequential letter between A and Z).

Collectively these sample files are referred to as Tools for Automated System Management (TASM). The procedures for installing TASM are as follows:

1. Create the following directory on the disk of your choice:

```
$ CREATE/DIR device:[TASM.COM]
```

2. Copy the files from the diskette to the [TASM.COM] directory using your favorite terminal emulator, Kermit, or networking product (e.g., Digital Pathworks).

   Using Kermit, for example

```
$ SET DEFAULT device:[TASM.COM]
$ KERMIT
KERMIT> SERVER              ! Put Kermit in Server Mode
[then issue switch sequence to attach to PC]
KERMIT> SEND B:*.*          ! Send all files
```

3. On your Digital system execute RENAME.COM to rename the files from their DOS names to their full OpenVMS names.

```
$ @RENAME
%RENAME-I-RENAMED, TASM$ROOT:[COM]FILE05A.TXT;1 renamed to
TASM$ROOT:[COM]SYCONFIG.COM;1
```

```
%RENAME-I-RENAMED, TASM$ROOT:[COM]FILE05B.TXT;1 renamed to
TASM$ROOT:[COM]SYLOGICALS.COM;1

%RENAME-I-RENAMED, TASM$ROOT:[COM]FILE05C.TXT;1 renamed to
TASM$ROOT:[COM]SYPAGSWPFILES.COM;1

%RENAME-I-RENAMED, TASM$ROOT:[COM]FILE05D.TXT;1 renamed to
TASM$ROOT:[COM]SYSECURITY.COM;1

...
$
```

4. Edit TASM$STARTUP.COM and configure the TASM$DSK logical to reflect the device you've selected to contain TASM

```
$ EDIT TASM$STARTUP.COM
$! TASM$STARTUP.COM
$!
$ DEFINE TASM$DSK DISK$USER2:
..
[Save the changes and exit to DCL]
```

5. Execute TASM$STARTUP with the SETUP parameter. This forces the remaining TASM directories and logical names to be created, as follows:

```
$ @TASM$STARTUP SETUP
...
%CREATE-I-EXISTS, TASM$COM already exists
%CREATE-I-CREATED, TASM$ROOT:[BACKUPS] created
%CREATE-I-CREATED, TASM$ROOT:[DOC] created
%CREATE-I-CREATED, TASM$ROOT:[SCRATCH] created
```

   After you execute TASM$STARTUP you can refer to all TASM directories by their respective logical names. For example, TASM$COM refers to the [TASM.COM] directory.

6. Edit your system startup files (e.g., STARTUP_PRODUCTS.COM) so that TASM$STARTUP is executed every time the system boots, as follows:

```
$ EDIT SYS$MANAGER:STARTUP_PRODUCTS.COM
...
...
$! Start TASM
$!
```

```
$ @device:[TASM.COM]TASM$STARTUP
..
[Save the changes and exit to DCL]
```

7. Execute TASM_SYMBOLS to define TASM symbols, as follows:

```
$ @TASM$COM:TASM_SYMBOLS
Defining TASM symbols. Enter "TASM" for HELP.
```

Also add the foregoing command to the login command procedures of users who are to access TASM (see chapter 11). After you define the symbols, issue the TASM command to see the symbols that have been defined, as follows:

```
$ TASM

The following TASM symbols are defined:

ACCT              Manage User Accounts
BCK               Perform Backup Functions
MAIL_COMPRESS     Compress VMS Mail Files
SHUT              Display Shutdown Menu
SYSDOC            Access On-line Site Guide
SYSGEN            Run SYSGEN Utility
SYSMAN            Run SYSMAN Utility
TASM              Display this help screen
UAF               Run AUTHORIZE Utility
```

Once you've installed TASM, refer to the following sections for the procedures on using each TASM component.

Chapter 5

The files referenced in chapter 5 are listed in the following table.

Procedure Name	*Diskette File Name*	*Function*
SYCONFIG.COM	FILE05A.TXT	Configure devices
SYLOGICALS.COM	FILE05B.TXT	Define logical names
SYPAGSWPFILES.COM	FILE05C.TXT	Install page/swap files
SYSECURITY.COM	FILE05D.TXT	Configure security
SYSTARTUP_VMS.COM	FILE05E.TXT	Main startup file
STARTUP_MOUNT.COM	FILE05F.TXT	Mount disks
STARTUP_NETWORK.COM	FILE05G.TXT	Start up network components

STARTUP_SETTERM.COM	FILE05H.TXT	Configure terminal devices
STARTUP_QUEUES.COM	FILE05I.TXT	Set up queues
STARTUP_SHARE.COM	FILE05J.TXT	Install images
STARTUP_SDA.COM	FILE05K.TXT	Analyze crash dump
STARTUP_PRODUCTS.COM	FILE05L.TXT	Start vendor products
STARTUP_APPLICATIONS.COM	FILE05M.TXT	Start in-house applications

The following files are sample system startup procedures that you can use as a guide for designing your startup procedures.

Chapter 6

The files referenced in chapter 6 are listed in the following table.

Procedure Name	*Diskette File Name*	*Function*
SHUTDOWN_MENU.COM	FILE06A.TXT	Shut down system

SHUTDOWN_MENU allows you to execute several commonly used shutdown sequences from a menu. With minor modifications you can customize menu options to reflect the needs of your site. The most important benefit of using this technique is that you can ensure that the procedures you define in the menu are used to shut down the systems. For example, the technique allows you to ensure that certain shutdown options are always selected (e.g., REMOVE_NODE).

The symbol SHUT is defined in TASM_SYMBOLS. It facilitates access to the SHUTDOWN_MENU. For example:

```
$ SHUT
SHUTDOWN MENU

1- RETURN       Return to previous menu
2- REBOOT       Shutdown now with immediate reboot
3- REBOOT5      Shutdown in 5 minutes, with immediate reboot
4- NORMAL       Shutdown now without reboot
5- NORMAL5      Shutdown in 5 minutes, without reboot
6- POWEROFF     Shutdown, prepare for power shutoff

Enter a number:
```

Chapter 9

The files referenced in chapter 9 are listed in the following table.

Procedure Name	*Diskette File Name*	*Function*
ACCT_ADDUSER.COM	FILE09A.TXT	Add new user account
ACCT_REMUSER.COM	FILE09B.TXT	Remove user account
ACCT_UAFRPT.COM	FILE09C.TXT	Produce UAF report
ACCT_MENU.COM	FILE09D.TXT	Menu to access ACCT functions

The examples presented in chapter 9 allow you to add and remove user accounts and to produce an account listing. The symbol ACCT is defined in TASM_SYMBOLS, which allows easy access to the ACCT menu. For example:

```
$ ACCT

MANAGE USERS MENU

1- RETURN       Return to previous menu
2- ADDUSER      Add a new user
3- REMUSER      Remove an existing user
4- UAFRPT       Produce a report of all existing users

Enter a number:
```

ADDUSER

First add template accounts using Authorize and specify all the defaults you want to use for that group of users. For example, to create a template for all executive accounts, issue the following command:

```
$ UAF
UAF> ADD TMPL_EXEC/UIC=[100,1]/ACCOUNT="EXECUTIV"/DEVICE=DSK$EXECUTIVE-
_UAF> /FLAGS=DISUSER/PRIV=(TMPMBX,NETMBX)/DEFPRIV=(TMPMBX,NETMBX)-
_UAF> /PWDLIFETIME="60-0"/OWNER="TEMPLATE ACCOUNT"
```

Next, modify ACCT_ADDUSER.COM to reflect the templates that you've added. You must specify the template name, description, default login device, and user identification code (UIC) group for each template, as follows:

```
$! ACCT_ADDUSER.COM
$! ...
$! Add your template names below (leave first entry blank)
$ tmpl_name = "//TMPL_EXEC/TMPL_BILL/TMPL_HR/TMPL_SHIP/TMPL_SYST/"
$! Define default login device, UIC group, and description
$ tmpl_dev = "//DSK$EXECUTIVE/DSK$BILLING/DSK$HR/DSK$SHIPPING/DSK$SYSTEMS/"
$ tmpl_grp = "//100/200/400/600/700/"
$ tmpl_desc = "//Executive/Billing/Human Resources/Shipping/Systems/"
```

Then invoke ACCT and select ADDUSER from menu. You will then get the following menu listing all the defined templates:

```
ADD USER ACCOUNT

1-RETURN        Return to previous menu
2-TMPL_EXEC     Executive
3-TMPL_BILL     Billing
4-TMPL_HR       Human Resources
5-TMPL_SHIP     Shipping
6-TMPL_SYST     Systems
```

Select a template that you want to use and enter the username, as follows:

```
Enter account template to use: 2

Last name: Simpson
First name: Howard
Middle initial: A

Owner      Username    UIC       Account    Privs    Pri  Directory

           TMPL_EXEC   [100,1]   EXECUTIV   Normal   4    Disuser

Choose a UIC Member number, excluding the ones listed above.
```

A list of existing users in the group you selected is displayed. Use this information to locate an available member number and enter it, as follows:

```
Account's UIC Member number: 2
```

The account and its default login directory are then automatically created, as follows:

```
%CREATE-I-CREATED, DSK$EXECUTIVE:[SIMPSONH] created
%UAF-I-COPMSG, user record copied
```

```
%UAF-I-RDBADDMSGU, identifier SIMPSONH value: [000100,000002] added
to rights data base

Username: SIMPSONH          Owner: HOWARD A. SIMPSON
Account: EXECUTIV           UIC: [100,2] ([EXECUTIV,SIMPSONH])
CLI: DCL                    Tables: DCLTABLES
Default: DSK$EXECUTIVE:[SIMPSONH]
LGICMD: LOGIN
Flags: DisUser
Primary days: Mon Tue Wed Thu Fri
Secondary days: Sat Sun
No access restrictions
Expiration: (none)          Pwdminimum: 6 Login Fails: 0
Pwdlifetime: 60 00:00       Pwdchange: (pre-expired)
Last Login: (none) (interactive), (none) (noninteractive)
Maxjobs: 0           Fillm: 100           Bytlm: 8192
Maxacctjobs: 0       Shrfillm: 0          Pbytlm: 0
Maxdetach: 0         BIOlm: 18            JTquota: 1024
Prclm: 2             DIOlm: 18            WSdef: 150
Prio: 4              ASTlm: 24            WSquo: 256
Queprio: 0           TQElm: 10            WSextent: 512
CPU: (none)          Enqlm: 100           Pgflquo: 10240
Authorized Privileges:
NETMBX TMPMBX
Default Privileges:
NETMBX TMPMBX

Account added.
                    Press [return] for main menu
```

REMUSER

REMUSER prompts you for the name of an account to delete. It archives the user's default directory (and all subdirectories) to a save set in the TASM$ARCHIVE directory. The same set is named username.SAV. Then the user's entire directory tree and user authorization file (UAF) record are deleted.

To delete a user, select REMUSER from the ACCT menu. When you are prompted for a username, enter the name of the account you want to delete, as follows:

```
Username to delete: SIMPSONH

Are you Sure you want to archive DSK$EXECUTIVE:[SIMPSONH...]*.*;* (Y/N): Y

%BACKUP-S-COPIED, copied DSK$EXECUTIVE:[SIMPSONH]LOGIN.COM;1
```

```
%BACKUP-S-COPIED, copied DSK$EXECUTIVE:[SIMPSONH]MAIL.MAI;1

%BACKUP-I-STARTDELETE, starting file deletion pass

Home directory for SIMPSONH has been deleted.

Are you Sure you want to remove the account SIMPSONH (Y/N): Y

%UAF-I-REMMSG, record removed from system authorization file
%UAF-I-RDBREMMSGU, identifier SIMPSONH value: [000100,000002] removed from
          rights data base

The account, SIMPSONH, has been completely removed.
```

UAFRPT

The UAFRPT option allows you to generate and optionally print a list of all user accounts. By default, the listing file is named SYSUAF.LIS and is located in the TASM$SCRATCH directory. For example:

```
Print Report (Y/N): N

%UAF-I-LSTMSG1, writing listing file
%UAF-I-LSTMSG2, listing file SYSUAF.LIS complete

$ TYPE TASM$SCRATCH:SYSUAF.LIS

Owner                Username      UIC         Account   Privs   Pri Directory

SYSTEM MANAGER       SYSTEM        [1,4]       SYSTEM    All     4  SYS$SYSROOT:[SYSMGR]
FIELD SERVICE        FIELD         [1,10]      FIELD     All     4  SYS$SYSROOT:[SYSMAINT]
                     TMPL_EXEC     [100,1]     EXECUTIV  Normal  4  Disuser
                     TMPL_BILL     [200,1]     BILLING   Normal  4  Disuser
                     DEFAULT       [200,200]             Normal  4  Disuser
NML$SERVER DEFAULT   NML$SERVER    [376,371]   DECNET    Normal  4  SYS$SPECIFIC:[NML$SERVER]
PHONE$SERVER DEFAULT PHONE$SERVER  [376,372]   DECNET    Normal  4  SYS$SPECIFIC:[PHONE$SERVER]
MAIL$SERVER DEFAULT  MAIL$SERVER   [376,374]   DECNET    Normal  4  SYS$SPECIFIC:[MAIL$SERVER]
DECNET DEFAULT       DECNET        [376,376]   DECNET    Normal  4  SYS$SPECIFIC:[DECNET]
                     TMPL_HR       [400,1]     HR        Normal  4  Disuser
DSN$NETWORK          DSN$SERVER    [500,1]               Normal  4  SYS$COMMON:[DSN.LOGS]
                     TMPL_SHIP     [600,1]     SHIPPING  Normal  4  Disuser
                     TMPL_SYST     [700,1]     SYSTEMS   Normal  4  Disuser

                          Press [return] for main menu
```

Chapter 10

The files referenced in chapter 10 are listed in the following table.

Procedure Name	*Diskette File Name*	*Function*
CUTTER.COM	FILE10A.TXT	Archive system log files

The purpose of CUTTER is to periodically close system log files (e.g., ACCOUNTNG.DAT, OPERATOR.LOG) and archive them to a temporary directory. To make it easier to identify archived files, the file names include a month and date time stamp. By default, files are archived to the directory defined by the TASM$ARCHIVE login name (defined in TASM$STARTUP), as follows:

```
$ @CUTTER

%COPY-S-COPIED, DISK$USER1:[SYSFILES]ACCOUNTNG_BIRCH.DAT;11 copied to
TASM$ROOT:[ARCHIVE]ACC_JUN_1995.DAT;11 (1 block)

%COPY-S-COPIED, SYS$COMMON:[SYSMGR]SECURITY.AUDIT$JOURNAL;5 copied to
TASM$ROOT:[ARCHIVE]SEC_JUN_1995.DAT;5 (1 block)

%COPY-S-COPIED, SYS$SYSROOT:[SYSERR]ERRLOG.SYS;1 copied to
TASM$ROOT:[ARCHIVE]ERR_JUN_1995.DAT;1 (1 block)

%COPY-S-COPIED, DISK$USER1:[SYSFILES]OPERATOR_BIRCH.LOG;75 copied to
TASM$ROOT:[ARCHIVE]OPER_JUN_1995.DAT;75 (2 blocks)
```

The command procedure MONTHLY_PROCESSING (see chapter 12) invokes CUTTER to force this process to occur each month.

Chapter 11

The files referenced in chapter 11 are listed in the following table.

Procedure Name	*Diskette File Name*	*Function*
DEFINE_SYMB.MAR	FILE11A.TXT	Macro code to define DCL symbols

This file is a VAX Macro example that demonstrates how to define DCL symbols from a compiled language. Procedures for using this file are presented in chapter 11.

Chapter 12

The files referenced in chapter 12 are listed in the following table.

Procedure Name	*Diskette File Name*	*Function*
SCHEDULER.COM	FILE12A.TXT	Main scheduler
DAILY_PROCESSING.COM	FILE12B.TXT	Daily tasks
WEEKLY_PROCESSING.COM	FILE12C.TXT	Weekly tasks
MONTHLY_PROCESSING.COM	FILE12D.TXT	Monthly tasks

These files comprise a sample automated batch scheduling system. Procedures for using these files are presented in chapter 12.

Chapter 13

The files referenced in chapter 13 are listed in the following table.

Procedure Name	*Diskette File Name*	*Function*
BCK_SETS.COM	FILE13A.TXT	Defines backup sets and attributes
BCK_NIGHTLY.COM	FILE13B.TXT	Main backup program
BCK_CALC_LABEL.COM	FILE13C.TXT	Calculate tape label to use
BCK_SELECT_FILE.COM	FILE13D.TXT	Select file to restore
BCK_RESTORE_FILE.COM	FILE13E.TXT	Perform file restore
BCK_MENU.COM	FILE13F.TXT	Main backup menu
BCK_ERRORS.COM	FILE13G.TXT	Backup error codes

Together these example procedures implement the basic backup, tape labeling, and retention strategies discussed in chapter 13.

To use these procedures you first have to organize your disks into backup sets. Sets are designed to allow you to back up several disks to a single, high-capacity medium (e.g., 4-mm or 8-mm tapes). If you want to perform unattended backups, assign only as many disks per set as will fit on a single tape.

For example, consider a system with the following devices:

```
DISK$USER1      0.5GB (disk drive)
DISK$USER2      0.5GB (disk drive)
DISK$SCR1       1.0GB (disk drive)
MUA0:           1.0GB (tape drive)
```

Because the capacity of the tape drive is limited to 1GB, you have to create at least two backup sets, as follows:

```
Set S01:        DISK$USER1, DISK$USER2
Set S02:        DISK$SCR1
```

Next, edit TASM$COM:BCK_SETS to reflect the backup sets that you've selected. Configure the symbols as follows:

BCK_NUMSETS	The number of backup sets you have defined
BCK_SETS	List of backup set names separated by slashes; set names must have exactly three characters each
BCK_TAPE	Tape drive to use for this backup set
BCK_OUTQUAL	Any special backup qualifiers you need
BCK_COUNT	Number of devices in the backup set
BCK_DEVICES	List of devices separated by slashes ("/")
BCK_SAVESETS	Names of savesets to use for each device

The values you establish for the BCK_DAILY, BCK_WEEKLY, BCK_MONTHLY and BCK_YEARLY symbols determine the backup schedule and the retention period.

The retention period for each backup type is as follows:

DAILY	1 month
WEEKLY	1 month
MONTHLY	1 year
YEARLY	2 years

Configure the following symbols to establish the type of backup to perform on each day of the month:

BCK_DAILY	The days of the week to perform daily backups; select one or more days (Mon–Sun)
BCK_WEEKLY	Day of the week to perform weekly backup; select one day of week (Mon–Sun)
BCK_MONTHLY	Week within month to perform monthly backup; select one week number (1–6)
BCK_YEARLY	Month within year to perform yearly backup; select one month of year (1–12)

For example:

```
$! BCK_SETS.COM
$! ...
$ bck_numsets == 2
$ bck_sets == "/S01/S02/"
$ bckset = p1
$ goto SET_'bckset'
$! ---------------------
$! Backup Disk Set "S01"
$! ---------------------
$SET_S01:
$ BCK_TAPE == "MUA0:"
$ BCK_OUTQUAL == "/BLOCK=40000/GROUP=10"
$ BCK_COUNT == 2
$ BCK_DEVICES == "/DISK$USER1/DISK$USER2/"
$ BCK_SAVESETS == "/IMAGEU1/IMAGEU2/"
$ BCK_DAILY == "MonTueWedThu"  ! Daily backups on these days
$ BCK_WEEKLY == "Fri"          ! Weekly backup on this day
$ BCK_MONTHLY == "2"           ! Monthly on this week number
$ BCK_YEARLY == "2"            ! Yearly during 2nd month of year
$ EXIT
$!
$! ---------------------
$! Backup Disk Set "S02"
$! ---------------------
$SET_S02:
$ BCK_TAPE == "MUA0:"
$ BCK_OUTQUAL == "/BLOCK=40000/GROUP=10"
$ BCK_COUNT == 1
$ BCK_DEVICES == "/DISK$SCR1/"
$ BCK_SAVESETS == "/IMAGES1/"
$ BCK_DAILY == "Mon"           ! Daily backups on these days
$ BCK_WEEKLY == "Tue"          ! Weekly backup on this day
$ BCK_MONTHLY == "2"           ! Monthly on this week number
$ BCK_YEARLY == ""             ! No Yearly backup
$ EXIT
```

If you need more backup sets, add additional entries using the same format.

The resulting backup schedule for backup set S01 is daily backups Monday through Thursday, weekly backups on Fridays, monthly backups on the second week of the month (on Friday), and a yearly backup each February. Once you have specified this information and given a specific date, the backup procedures can always calculate exactly what type of backup should be performed and the tape label that should be used.

This process is automated by the procedure BCK_CALC_LABEL.COM. Before a backup begins, the main backup program (BCK_NIGHTLY) invokes

CALC_LABEL to determine if a backup is scheduled for that day and if so what label should be used. CALC_LABEL checks the symbols that define the schedule (e.g., BCK_DAILY). If there is no backup scheduled, it does not return a label. If there is a backup scheduled, CALC_LABEL generates one by concatenating the backup set name, the type of backup, and a sequential number.

The first three characters of the label contain the backup set name. The next character is the backup type abbreviated as D, W, M, and Y. The sequential number is set depending on the backup type. For yearly backups it is set to 1 for odd years and 2 for even years. For monthly backups it is set to the month number (e.g., Jan = "1," Feb = "2," etc.) For weekly and daily backups, the sequential number is set to the week or day number, as follows:

	S	*M*	*T*	*W*	*R*	*F*	*S*	*Week #*
	1	2	3	4	5	6	7	1
	8	9	10	11	12	13	14	2
Day #	15	16	17	18	19	20	21	3
	22	23	24	25	26	27	28	4
	29	30	31	32	33	34	35	5
	36	37						6

For example, May 1, 1993 is day number 7, week number 1. May 31 is day number 37, week number 6.

May 1993	*S*	*M*	*T*	*W*	*R*	*F*	*S*
							1
	2	3	4	5	6	7	8
	9	10	11	12	13	14	15
	16	17	28	19	20	21	22
	23	24	25	26	27	28	29
	30	31					

Given the following dates and the schedule for backup set S01 presented earlier, BCK_CALC_LABEL would generate labels as follows:

```
May 1, 1993          No Backup
May 7, 1993          S01M5
May 14, 1993         S01W3
May 25, 1993         S01D31
May 31, 1993         S01D37
```

Once you have configured BCK_SETS, you should organize all the backup media that you will need, initialize each tape with the required label, and attach a written label to each tape. You can generate a backup schedule to see

exactly which tapes will be requested for each day of the month (see the next section).

After you have all your tapes prepared you are ready to initiate a backup. To start a backup, you have to submit BCK_NIGHTLY as a batch job and pass it through several parameters, including the following:

P1	The backup set to process
P2	The scheduled backup date, or "" for the current date
P3	"PRE" if you just want to verify that the proper tape is loaded in the tape drive

To simplify this process you can create a command procedure such as BCK_NIGHTLY_SUB.COM, as follows:

```
$ TYPE BCK_NIGHTLY_SUB.COM
$! BCK_NIGHTLY_SUB.COM
$!
$! P1 - Backup set
$! P2 - Scheduled backup date
$! P3 - If TOMORROW then submit for next day
$!
$ submit/noprint/keep/log=TASM$BACKUPS:BCK_NIGHTLY_PRE.LOG -
     /NAME="BCK_NIGHTLY_PRE"/notify/user=system/param= -
     ("''p1'","''p2'","PRE")/after="''P3'15:30" TASM$COM:BCK_NIGHTLY
$ submit/noprint/keep/log=TASM$BACKUPS:/notify/user=system/param= -
     ("''p1'","''p2'")/after="''P2'23:00" TASM$COM:BCK_NIGHTLY
$ exit
```

Then to submit the backup, issue the following DCL command:

```
$ @BCK_NIGHTLY_SUB S01 ""
```

The foregoing procedure submits BCK_NIGHTLY with the "PRE" option for 3:30 p.m. and without the "PRE" option for 11:00 p.m. When the PRE job runs, it checks the tape drive to see if the proper tape is loaded. If no tape or the wrong tape is loaded, all terminals enabled to receive tape requests (REPLY/ENABLE=TAPES) receive a request message such as the following:

```
%%%%%%%%%%% OPCOM 11-JUN-1993 13:29:44.32 %%%%%%%%%%%
Request 1, from user SYSTEM on BIRCH
BATCH_18, Wrong volume mounted (S01D13), mount S01W2 in MUA0: and use
REPLY/TO.
```

This alerts you to put the correct tape in before you leave for the day. Once the correct tape is in the drive, issue the following command to respond to the request:

```
$ REPLY/TO = 1

$ 13:31:18.24, request 1 was completed by operator _BIRCH$OPA0:
```

The PRE job then checks the tape again and sends the following message if the correct tape is loaded:

```
%%%%%%%%%%% OPCOM 11-JUN-1993 13:53:05.76 %%%%%%%%%%%
Message from user SYSTEM on BIRCH

BATCH_25, Tape: S01W2 is loaded and ready for tonight's backup.
Job BCK_NIGHTLY_PRE (queue BIRCH_GENERAL, entry 25) completed
```

Finally, at 11:00 p.m. the backup job begins and performs the backup. During the backup, BCK_NIGHTHLY creates a backup listing file in the TASM$BACKUPS directory that lists every file that is copied to backup media. Additionally, a log file is also written in the TASM$BACKUPS directory.

To force backups to be submitted automatically, you should invoke BCK_NIGHTLY_SUB from an automated scheduling system such as DAILY_PROCESSING.COM (see chapter 12).

The following sections discuss the other components of the backup examples.

BCK_MENU

The symbol BCK is defined by TASM_SYMBOLS to invoke BCK_MENU.COM, which allows you to select a number of backup options from a menu, as follows:

```
$ BCK

                        BACKUP/RESTORE MENU

     1-RETURN        Return to previous menu
     2-RESUBMIT      Resubmit BCK_NIGHTLY
     3-RESTORE       Restore a file from backups
     4-SCHED         Print a backup schedule

Enter a number:
```

RESUBMIT

Invokes BCK_NIGHTLY_SUB to manually resubmit a backup for the current day.

RESTORE

Use the RESTORE option from the backup menu to restore a file from backups. The RESTORE option executes the command procedure BCK_SELECT_FILE. You are then prompted for various information to locate the backup from which you want to restore. BCK_SELECT_FILE then submits BCK_FILE_RESTORE as a batch job, passing it the file name and tape label you selected. The restore procedure copies the restored file to TASM$RESTORE:[*] and notifies you when the restore has completed.

The following is an example restore of [SYS0.SYSCOMMON.SYSEXE]SYSUAF.DAT from DISK$USER1:

```
[select RESTORE from Backup Menu]

1- S01
2- S02
Select a backup set (1-2): 1
```

First choose the backup set that contains DISK$USER1, as follows:

```
1- DISK$USER1- VAXVMSV055 (BIRCH$DUA0:)
2- DISK$USER2- VAXVMST060 (BIRCH$DUA1:)
Select disk by number (1-2): 1
```

Then select the proper disk. BCK_SELECT_FILE then lists all listing files that exist for that disk.

```
Searching all DISK$USER1 listing files...

The following is a list of backup listing files that exist. They are
sorted by date of backup (latest backups listed first).

Press "N" for the next page
Press "P" for the previous page
Press <RETURN> to return to the previous menu

1-S01W2_IMAGEU1.LIS;1 1993-06-11 13:57:46.4

Enter choice # (1-10), P or <RETURN>: 1
```

If there were many from which to choose, you could use N and P to see the next or previous page of listing files. In this case, there is only one choice (S01W2_IMAGEU1).

```
Listfile = TASM$BACKUPS:S01W2_IMAGEU1.LIS;1
```

Enter the filename you wish to restore. Enter the filename in the following format:

```
[directory.subdir.]filename.type
e.g., [USER.SMITH]LOGIN.COM
```

You may leave out any part of the filename, but the less specific you are the more choices you will have to select from.

```
e.g., LOGIN.COM

Enter filename: SYSUAF.DAT
```

Now, enter the filename you want to restore. In this case SYSUAF.DAT. BCK_SELECT_FILE scans the listing file you selected for all occurrences of that file name. Again, use N or P to scroll from one page to another if there are many entries.

```
1- [SYS0.SYSCOMMON.SYSEXE]SYSUAF.DAT;1 32 17-OCT-1991
2- [SYSFILES]SYSUAF.DAT;1 42 17-OCT-1991 13:36

Enter choice # (1-10), N, P or <RETURN>: 1
```

In this case choose selection 1 because it corresponds exactly to the file you want to restore. BCK_SELECT_FILE summarizes the parameters it will use to restore the file and submits BCK_RESTORE_FILE as a batch job, as follows:

```
Tape drive to use [MUA0:]:
label = S01W2
saveset = IMAGE
file = [SYS0.SYSCOMMON.SYSEXE]SYSUAF.DAT;1

Job BCK_RESTORE_FILE (queue BIRCH_GENERAL, entry 15) started on
BIRCH_GENERAL
```

When BCK_RESTORE_FILE is complete, it sends the following broadcast message indicating that the restore was successful:

```
Reply received on BIRCH from user SYSTEM at _BIRCH$OPA0: 11:27:30
File: [SYS0.SYSCOMMON.SYSEXE]SYSUAF.DAT;1 restored to
TASM$RESTORE:[SYS0.SYSCOMMON.SYSEXE]
$ DIR TASM$RESTORE:[SYS0.SYSCOMMON.SYSEXE]/SIZE=ALL/DATE

Directory TASM$RESTORE:[SYS0.SYSCOMMON.SYSEXE]

SYSUAF.DAT;1 32/33 17-OCT-1991 13:36:51.96

Total of 1 file, 32/33 blocks.
```

Once you believe that you have restored the correct file, copy it to its production directory.

SCHED

Use the SCHED option to generate a backup schedule for a specific backup set and month. For example:

```
[select SCHED from backup menu]

Enter a date (DD-MM-YYYY) that is in the month that you want a backup
schedule for [CURRENT MONTH]: 01-May-1993
Enter Backup Set [S01]:
Print it (Y/N)? [N]:

$ TYPE TASM$SCRATCH:BCK_SCHED_S01.RPT

                        Schedule for MAY 1993
                           Backup Set: S01

Sat    , 1-MAY-1993 :
Sun    , 2-MAY-1993 :
Mon    , 3-MAY-1993 : S01D9
Tue    , 4-MAY-1993 : S01D10
Wed    , 5-MAY-1993 : S01D11
Thu    , 6-MAY-1993 : S01D12
Fri    , 7-MAY-1993 : S01M5
Sat    , 8-MAY-1993 :
Sun    , 9-MAY-1993 :
Mon    , 10-MAY-1993: S01D16
Tue    , 11-MAY-1993: S01D17
Wed    , 12-MAY-1993: S01D18
Thu    , 13-MAY-1993: S01D19
Fri    , 14-MAY-1993: S01W3
Sat    , 15-MAY-1993:
```

```
Sun    , 16-MAY-1993:
Mon    , 17-MAY-1993: S01D23
Tue    , 18-MAY-1993: S01D24
Wed    , 19-MAY-1993: S01D25
Thu    , 20-MAY-1993: S01D26
Fri    , 21-MAY-1993: S01W4
Sat    , 22-MAY-1993:
Sun    , 23-MAY-1993:
Mon    , 24-MAY-1993: S01D30
Tue    , 25-MAY-1993: S01D31
Wed    , 26-MAY-1993: S01D32
Thu    , 27-MAY-1993: S01D33
Fri    , 28-MAY-1993: S01W5
Sat    , 29-MAY-1993:
Sun    , 30-MAY-1993:
Mon    , 31-MAY-1993: S01D37
```

You should generate the schedules several months in advance so that you know exactly when you need each tape.

Chapter 14

The files referenced in chapter 14 are listed in the following table.

Procedure Name	*Diskette File Name*	*Function*
SYSDOC.COM	FILE14A.TXT	On-line site guide

SYSDOC is a tool for creating and accessing on-line site documentation (e.g., policies, procedures, contact names, configuration information). Before you can use SYSDOC you must create directories that contain documentation text files. When you invoke SYSDOC, the directory and file names are presented as menu options, allowing you to traverse up and down directories and to view documents.

SYSDOC expects all directories to be created under the directory defined by the logical name TASM$DOC. For example, you might want to store documentation relating to contact names, applications, and emergency procedures. First create a directory for each of subject, as follows:

```
$ SET DEFAULT TASM$DOC
$ DIR
%DIRECT-W-NOFILES, no files found
```

```
$ CREATE [.CONTACTS]/DIR/PROT=(W:RE)
$ CREATE [.APPLICATIONS]/DIR/PROT=(W:RE)
$ CREATE [.EMERGENCY]/DIR/PROT=(W:RE)

$ DIR

Directory TASM$ROOT:[DOC]

APPLICATIONS.DIR;1 CONTACTS.DIR;1 EMERGENCY.DIR;1

Total of 3 files.
```

Then create as many documents as you want within each directory. When you name a document file, the file name must be DOC, but the file extension can be whatever you want.

```
$
$ CREATE [.CONTACTS]DOC.DIGITAL
Digital Contacts

Service: (800)-354-9000
Sales: (203)-555-2345 Bill Smith
DECdirect: (800)-344-4825
```

Then invoke SYSDOC and select CONTACTS, as follows:

```
$ SYSDOC

                        Online Site Guide

1- APPLICATIONS
2- CONTACTS
3- EMERGENCY

Enter Choice #: 2
```

The documents in the CONTACTS directory are then displayed in a menu. Select the document you want to edit or view.

```
                        Online Site Guide

1- DIGITAL     Digital Contacts

Enter Choice #: 1
E- Edit
T- Type
```

```
P- Print
Action: T

Digital Contacts

Service:    (800)-354-9000
Sales:      (203)-555-2345 Bill Smith
DECdirect:  (800)-344-4825
< Press RETURN to continue >
```

At any menu just press the <RETURN> key to return to the previous level.

Chapter 15

The files referenced in chapter 15 are listed in the following table.

Procedure Name	*Diskette File Name*	*Function*
CHECK_QUEUES.COM	FILE15A.TXT	Monitor queues
CHECK_PROCESSES.COM	FILE15B.TXT	Monitor processes
CHECK_PROCESSES.DAT	FILE15C.TXT	Define processes to watch
CHECK_PROCESSES.FDL	FILE15D.TXT	Define format to load indexed file from .DAT
CHECK_DISKS.COM	FILE15E.TXT	Monitor disks
CHECK_DISKS.DAT	FILE15F.TXT	Define disks to watch

You can use the sample procedures in this section to monitor the status of queues, system processes, and disks. All procedures execute as detached processes that send operator request messages in the event a problem is detected. To receive these messages add the following command to your LOGIN.COM:

```
$ REPLY/ENABLE=CENTRAL    ! Requires OPER privilege
```

The following sections describe each procedure in more detail.

CHECK_QUEUES

CHECK_QUEUES uses the Get Queue Information lexical function (F$GETQUI) to monitor the status of all print and batch queues. By default, every 5 minutes CHECK_QUEUES verifies that all queues are started. If malfunctioning queues are detected, CHECK_QUEUES sends an operator request message to the CENTRAL class.

You can change the polling interval by adjusting the interval symbol defined in CHECK_QUEUES.COM. To start CHECK_QUEUES at system startup time, include the following line in your system startup procedures (e.g., STARTUP_PRODUCTS.COM):

```
$@TASM$COM:CHECK_QUEUES.COM
```

Add the line after the line that executes TASM$STARTUP so that all required logical names are defined.

CHECK_PROCESSES

CHECK_PROCESSES uses the Get Job Information lexical function (F$GETJPI) to monitor the status of selected system processes. By default, every 30 minutes CHECK_PROCESSES verifies that selected processes exist and are running under the proper usernames. If CHECK_PROCESSES detects the absence of a process or a process executing under the wrong username, it sends an operator request message to the CENTRAL class.

You can change the polling interval by adjusting the "interval" symbol defined in CHECK_PROCESSES.COM.

Define the processes and usernames that CHECK_PROCESSES monitors by creating TASM$COM:CHECK_PROCESSES.DAT, using the following format:

```
SWAPPER          /              /00/
ERRFMT           /SYSTEM        /00/
OPCOM            /SYSTEM        /00/
AUDIT_SERVER     /AUDIT$SERVER  /00/
JOB_CONTROL      /SYSTEM        /00/
NETACP           /DECNET        /00/
EVL              /DECNET        /00/
REMACP           /SYSTEM        /00/
<Check_Disks>    /SYSTEM        /00/
<Check_Queues>   /SYSTEM        /00/
```

To start CHECK_PROCESSES at system startup time, include the following line in your system startup procedures (e.g., STARTUP_PRODUCTS.COM):

```
$@TASM$COM:CHECK_PROCESSES.COM
```

Add this line after the line that executes TASM$STARTUP so that all required logical names are defined.

The first time CHECK_PROCESSES executes, it converts the entries from CHECK_PROCESSES.DAT into an indexed file (CHECK_PROCESSES.IDX) using the included file definition language file (CHECK_PROCESSES.FDL). The indexed file is then used for the life of the detached process. If you make changes to the text data file and you want to make them effective, you must manually stop the detached process and restart it.

CHECK_DISKS

CHECK_DISKS uses the Get Device Information lexical function (F$GETDVI) to monitor disk space and disk errors. By default, every 15 minutes it verifies that every disk has at least 50,000 free blocks and that error counts have not incremented. When CHECK_DISKS detects disk space below 50,000 blocks or a single disk error (a WARNING state), it sends an operator request message to the CENTRAL class. If disk space drops below 10,000 blocks, or 10 or more disk errors are detected (a CRITICAL state), CHECK_DISKS sends a REPLY message directly to selected usernames.

To adjust thresholds, polling interval, or the destination of messages, modify the following symbols in CHECK_DISKS.COM:

```
$! CHECK_DISKS.COM
$! ...
$ freeblock_warning = 50000          ! Disk space WARNING threshold
$ freeblock_critical = 10000         ! Disk space CRITICAL threshold
$ disk_err_warning = 1               ! Disk error WARNING threshold
$ disk_err_critical = 10             ! Disk error CRITICAL threshold
$ interval = "00:15:00"              ! polling interval
$ warning_class = "(central)"        ! Opcom class to reply to for warnings
$ critical_list = "(system)"         ! Usernames to notify if critical
```

When CHECK_DISKS is started, it determines which disks to monitor by reading the data file CHECK_DISKS.DAT. Before you start CHECK_DISKS, edit this file so that each disk you want to monitor is listed on a separate line, as follows:

```
DISK$USER1
DISK$USER2
```

To start CHECK_DISKS at system startup time, include the following line in your system startup procedures (e.g., STARTUP_PRODUCTS.COM):

```
$@TASM$COM:CHECK_DISKS.COM
```

Add this line after the line that executes TASM$STARTUP so that all required logical names are defined.

Chapter 16

The files referenced in chapter 16 are listed in the following table.

Procedure Name	*Diskette File Name*	*Function*
MAIL_COMPRESS.COM	FILE16A.TXT	Compress all mail files on a disk

Procedures for using this file are presented in chapter 16.

Chapter 17

The files referenced in chapter 17 are listed in the following table.

Procedure Name	*Diskette File Name*	*Function*
VMS$PASSWORD_POLICY.COM	FILE17A.TXT	Password filter
CAPTIVE_MENU.COM	FILE17B.TXT	Captive login procedure

Procedures for using these files are presented in chapter 17.

Chapter 18

The files referenced in chapter 18 are listed in the following table.

Procedure Name	*Diskette File Name*	*Function*
BENCH.COM	FILE18A.TXT	Gather resource usage statistics for an activity
HARD.MAR	FILE18B.TXT	Macro code to get hard page fault count

Procedures for using these files are presented in chapter 18.

Appendix E

The files referenced in this appendix are listed in the following table.

Procedure Name	*Diskette File Name*	*Function*
TASM$STARTUP.COM	FILE0FA.TXT	Define TASM logical names
TASM_MENU_ABORT.MSG	FILE0FB.TXT	Error message file
TASM_MENU_INPERR.MSG	FILE0FC.TXT	Error message file
TASM_SYMBOLS.COM	FILE0FD.TXT	Define TASM symbols
TASM_MENUSYM.COM	FILE0FE.TXT	Define symbols used by command procedures
RENAME.COM	RENAME.COM	Rename all files on this diskette to their full names

Procedures for using these files are discussed at the beginning of this chapter.

INDEX

Access, controlling system, 245-253.
See also Passwords
captive accounts and, 252-253
password content and length and, 246-251
restrictions for access and, 251-253
Access control list (ACL), 132, 137
Access control list (ACL)-based protection, 257-263
access control entries (ACEs) and
default, 262
identifier, 260-261
options of, 264
security, 262-263
establishing, 259-260
identifier attributes and, 259
right identifiers and, 258-259
Account defaults, 134-135
Account name, 134
ACCOUNTING utility, 291-292
Accounts, user. *See* User accounts, managing
Ancillary control process (ACP) caches, adjusting, 319-320
Audit reports, generating, 283-283
Auditable OpenVMS events, 282
Authorization database for user accounts, 127-129
AUTOGEN utility, 84, 327-332
cluster-common parameters and, establishing, 328, 329
feedback and, 330-332
modes of, 330
invoking, 329, 330
MODPARAMS.DAT entry formats of, 327
page, swap, and dump file entries and, 328
phases of, 329
system parameter entries and, 327-328
Automating system management, 176-185
customized error handling and, 178-179
default error handler and, 177-178
error handling and, 176-179
introduction to, 176
job schedulers and, 185
restart handling and, 179-181
batch job symbols for, 180
scheduling batch jobs and, 182-184
synchronizing batch jobs and, 181-182
tools for, 184-185
AWSA. *See* Working set adjustment, automatic

Backup window, 188
Backups, performing, 186-207
automatic tape handling and, 198-199
command examples for, 207
command qualifiers and, 196-197
cumulative incremental type of, 191-192
differential type of, 192
frequency of, 188-189
hardware failure and, 186-187
implementation of, 199-206
full, 200
full image restore with incremental backups, 202
incremental, 200-201
stand-alone, 192, 201-202
inconsistent backup risk reduction and, 193
input qualifiers and, 197

Backups, performing, *continued*
introduction to, 186-187
locating backup media and, 205, 206
media labeling and, 204-205
OpenVMS and, 206-207
output qualifiers and, 198
recovery time considerations and, 189
requirements for, 187-190
archive, 189
frequency, 188-189
issues to be considered for, 188
off-site rotation, 190
recovery time, 189
window, 188
risks of on-line backups and, minimizing, 193
software failures and, 187
stand-alone type of, 194
strategy development and, 194, 195
samples of, 195
tape handling commands and, 199
techniques for, 190-194
file, 191-192
image, 190
mode, 192
physical, 190-191
tools for, 194, 195-199
user authorization file (UAF) and, 206
user errors and, 187
volume shadowing and, 203-204
Batch jobs
monitoring, 289-290
scheduling, 182-184, 185
synchronizing, 181-182
system startup procedures and, 64
Batch queues, 105
characteristics of, 122
load-balancing, 120
names of, default logical, 120
Boot device, configuring, 30-35
boot control flags and, 31, 38
dedicated console subsystems and, 32-33
editing boot command files and, 33-35
nondedicated console systems and, 31-32
sequence and, 35
sources to boot from and, 30
Boot methods, 35-38
alternative system-disk boot and, 38
creating a, 43-44
automatic boot and, 36
automatic rebooting and, 72, 73
conversational boot and, 37
startup problems and, 47
default boot and, 37
manual boot and, 36-37
REBOOT_CHECK and, 75
stand-alone backup boot and, 37-38
Booting and various system bus types, 372-389
Booting OpenVMS, 25-44. *See also* System startup procedures
boot methods and, 35-38
building stand-along backup kits and, 43
configuring a boot device and, 30-35
console subsystems and, 26-30
control settings and, 29
creating alternative system disks and, 43-44
interrupt priority C (IPC) and, 42-43
introduction to, 25-26
minimal startup and, 39-40
password checking and, bypassing, 40-41
Snapshot facility and, 64-66
startup and, bypassing, 40
SYSBOOT options and, 39-42
adjusting SYSGEN parameters and, 41-42
reverting to DEFAULT parameters and, 42
reverting to old parameters and, 42
Bridge and Ethernet, 337-338

Capacity, expanding, 324
Captive accounts, 252-253
CHECK_DISKS and disk space management, 158
Cluster hang, 73
Configurations. *See* Site configurations; System configurations; individual operations
Console logging and system startup procedures, 62-63
Console subsystems, 26-33
  dedicated systems and, 26, 27, 32-33
  nondedicated systems and, 27
  processors with nondedicated type of, 31-32
  system restart control and, 29-30
    settings for, 29
  terminal and, 27-28
  terminal control and, 28, 29
Console summary, system, 372-389
Consolidated On-Line Documentation (OLD), 216
Contact name and addresses, 367-368
CPU dependency, estimating, 298-300
CPU load balancing, 304
CPU tuning, 321-323
  decompress system libraries and, 323
  process scheduling and, 321-322
  priority boosting and, 322-323
  quantum and, 323
Crash dump. *See also* Dump file, establishing common
  performing a, 82-63
  sizing the system for a, 83-85
    AUTOGEN and, 84-85
    SYSGEN and, 85
  when file does not exist for a, 85
CUTTER and disk space management, 158

Datalink address, 334, 335, 336, 337, 338
  LAT multicast and, 352
  terminal server and where to find the, 355
DCL. *See* Digital Command Language
DECnet
  addressing, 340, 341
  applications of, 341
  assigning addresses and host names for, 344
  components of, 340-343
  configuring node of, 344-348
  end nodes and, 341
  Internetwork example of, 342
  Level 1 (L1) routers and, 341, 342-343
  Level 2 (L2) routers and, 343
  management of, 340, 344-351
  network security and, 279-280
  shutting down, 348
  starting, 348
  troubleshooting, 349-351
DECUS. *See* Digital Equipment Corporation Users Society
Device protection and system management, 263, 264, 265, 266
Digital Command Language, 20. *See also* Login command procedures
  account management procedures and, 140
  BENCH.COM and, 292-294, 301, 324, 325
  console logging and, 62-63
  CUTTER and disk space management and, 158
  error severity levels in, 177
  mail management compression and, 242-243
  severity levels of, 177
  shutdown parameters of, 81, 87
  SYSTARTUP_Vxx.COM and, 53
Digital Corporation and VAXsim Plus, 227-228
Digital Customer Support Centers (CSCs), 219-221

Digital documentation for getting help, 210-216
Digital Equipment Corporation Users Society, 221-223
address for, 367
DECUServe and, 223
local users groups (LUGs) of, 221
master's lists and, 223
national symposia of, 222-223
software library of, 223
special interest groups (SIG) of, 221-222
Digital Multi-Datacenter Facility (MDF) product, 15
Digital software licenses, 91-92
layered products and, 91-92
system integrated products (SIPs) and, 91
transferability of, 92
Digital support and getting help, 216-221
centers for, 219-221
layered product support of, 218
media and documentation distribution service of, 218-219
self-maintenance and, 219
software for, 216-218
Directory, default, 16-17
Disaster planning and site configuration
backup site and, 14
distributed site and, 14-15
hot site and, 14
Disk defragmentation, 316-318
Disk I/O load balancing, 303
Disk input-output dependency, estimating, 297-298
Disk logical structure on OpenVMS system, 19
Disk monitoring, 228-229
Disk operation and system resource dependencies, 287
Disk organization for system configuration, 16-17
Disk space management, 142-158
archiving system log files and, 157-158
backup strategy of, 142-143
CHECK_DISKS as tool for, 158
cluster size and, 143
CUTTER as tool for, 158
expiration dates and, using file, 152-156
establishing volume retention and, 155-156
file version limits and, 145-147
defaults and, 146-147
establishing, 147
issues to consider for, 142
integrity monitoring and, 228-229
introduction to, 142
OpenVMS purge utility and, 143-145
quotas and, 148-152
enabling, 148-149
entry adding for, 150-151
entry managing and, 149-151
entry removing for, 151
rebuilding, 149
soft disks and, 152
tools for, 148
tools of, 158
volume rebuild and, 156-157
volume retention and, establishing, 155-156
Diskette enclosed, contents of, 394-418
Disks, creating alternative system, 43-44
Dump file. *See also* Crash dump
common, establishing a, 85-86
sizing, 83-85
using page file as, 85

Electrical attributes needed, 9-11
definition of, 7
electromagnetic interference (EMI) and, 11
electrostatic discharge and, 11
power basics and, 9-10

Emergency power off (EPO) switch, 12, 13
Emergency procedures for system shutdown, 69, 82-86
establishing common dump file as an, 85-86
OPCCRASH as an, 82
page file as system dump file as an, 85
performing a crash dump as an, 82-83
sizing the system dump file as an, 83-85
Environmental attributes needed, 11-12
corrosion and contamination and, 11-12
definition of, 7
humidity and, 11
temperature and, 11
ERASE_ON_DELETE, 266
Error handling, 176-179
customizing, 178-179
default, 177-178
severity levels of, DCL, 177
system startup and, 47-48, 61
Errors, monitoring for, 225-228
Ethernet
basics, 333, 334-340
addressing, 334, 335
cable types for, 336
interconnects and VMScluster configurations and, 22-23
LAT packet contents and, 357
network of, example of, 335
speed of, 358
terminology of, 336-340

Faults, hard and soft, 296-297
File backups, 191-192
File expiration dates, 152-156
establishing volume retention and, 155-156
selection commands for, 156
File object protection, 266, 267, 268-271
default file ownership and, 268, 269
default file protection and, 269-270, 271
defaults and commands of, 270
file access modes and, 269
File protection and login command procedures, set default, 164
File volatility, 191
backup strategy and, 194
differential backup and, 192
Form printing, 124-126
creating forms for, 124-125
Free page list, 295

Global sections, 308
Global symbols for login command procedures, defining, 163

Hard page faults and tuning memory, excessive, 305-311
establishing page file sizes and, 309, 310-311
image activation and, 305-308
image file I/O and, 308-309
page file I/O and, 309, 310
Hardware errors and integrity monitoring, 225-228
ANALYZE/ERROR and, 226-227
SHOW ERROR and, 226
VAXsim Plus and, 227-228
Hardware failure and performing backups, 186-187
Digital RA series disk annual failure rates and, 187
Help, getting, 208-224. *See also* Digital Equipment Corporation Users Society
CD-ROM documentation and, 216
Digital documentation and, 210-216
Digital Equipment Corporation Users Society (DECUS), 221-223

Help, getting, *continued*
Digital support and, 216-221
centers for, 219-221
layered product support of, 218
media and documentation distribution service of, 218-219
self-maintenance and, 219
software of, 216-218
help message utility (MSGHLP) and, 213
introduction to, 208
libraries for, 212
OpenVMS help text files and, 213-214
OpenVMS help utility (VMSHELP) and, 210-211
OpenVMS hints and, 211-212
OpenVMS manuals and, 214-216
version 5.0 versus 6.0 and, 215
site guide for, developing, 208-209
software patches and, 217
software support service (SSS) and, 216-218
electronic support and, advanced, 217
telephone support and, 217
HIGHWATER_MARKING, 265

Image backups, 190
Image file I/O and tuning memory, 308-309
Integrity monitoring, system, 225-234
disks and, 228-229
hardware errors and, 225-228
ANALYZE/ERROR and, 226-227
SHOW ERROR and, 226
VAXsim Plus and, 227-228
introduction to, 225
Operator Console Messenger (OPCOM) and, 233-234
procedures example of, 234
processes for, 229-233
tools for, 234
Interactive jobs, monitoring, 290-295
ACCOUNTING utility and, 291-292
Get Job/Process Information function and, 292-295
SHOW PROCESS and, 290-291
I/O, tuning, 316-321
ancillary control process (ACP) caches and, 319-320
disk defragmentation and, 316-318
virtual I/O cache and, 320-321
volume characteristics and, 318
Isolation transformer, 10

LAN. *See* Local area network
LAT. *See* Local area transport
LGI_ prefix and security-related system parameters, 275, 276, 277
Libraries
decompressing system and, 323
device control and, 121, 122-124
License management, 89-102
costs of, 92-95
Digital software licenses and, 91-92
installing, 99-100
introduction to, 89-91
Product Authorization Key (PAK) for, 89-91, 99-101
sample of, 90
types of, 92-95
ClusterWide kind of, 93-94
concurrent use kind of, 94
personal use kind of, 94-95
traditional kind of, 92
License management facility (LMF), 95-102
commands of, 98
database of, 95-96
Product Authorization Key (PAK) for, 99-101
restricting product access for, 101-102
tools of, 97-99
Load balancing, 302-304
activities included in, 302

Load balancing, *continued*
CPU and, 304
disk I/O and, 303
memory and, 302-303
Local area network, 351
DECnet and, 340
Ethernet and, 336, 337, 338, 339
Local area transport
access restriction with groups of, 364
components of, 351, 352-353
configuring groups of, 363-365
example of, 365
port type of, 364
server type of, 364
service node type of, 363, 364
configuring terminal server ports of, 359-360
connection basics for, 353
managing, 351-366
network example of, 352
printer device configuration and, 114-116
service nodes (hosts) of, managing, 351, 360-363
assigning service names and, 360-361
setting LAT node name and, 360
starting LAT and, 361-362
stopping LAT and, 362-363
terminal servers and, managing, 351, 354-359
configuration of, initial, 356-359
connecting to console port of, 354-356
software loading and, 354
troubleshooting, 365-366
Logical names for system configuration, 18, 20
Login command procedures, 159-175
captive type of, 174
cluster type of, 165-166
compiled language for, 171-173
DCL prompt for, setting, 163
disable interrupts and, 161
establishing menus for, 173-174
execute mode-specific and, 162
file protection for, setting default, 164-165
function keys for, defining, 163
global symbols for, defining, 163
group type of, 165
introduction to, 159
menus for, establishing, 173-174
printer for, assigning default, 164
reducing time for, 169-173
where to focus for, 170
sequence for, 160
system and, 160-168
template for, 166-168
terminal type for, setting, 162-163
user and, 135, 168

Mail Management, 235-243
attributes of, user accounts and, 135
database compression for, 237
distribution lists and, 240-241
introduction to, 235
logical names and, 239-240
message count synchronization and, forcing, 240
profile viewing, adding, and removing and, 238-239
system profile database and, 235-236
tools for, 242-243
user mail database and, 237-238
using as management tool, 241-242
Management of system, automating. *See* Automating system management
Media labeling and backups, 204-205
Memory operation and system resource dependencies, 287
estimating, 295-297
free page list and, 295
load balancing and, 302-303
modified list and, 295

Memory operation and system resources, *continued*
page fault and, 296-297
page replacement and, 295
summary chart of, 296
working set list and, 295
Memory tuning, 305-316. *See also* Hard page faults and tuning memory, excessive; Soft page faults and tuning memory, excessive
hard page faults and, excessive, 305-311
introduction to, 305
reclamation and, 314-316
soft page faults and, excessive, 311-314
Modified page writer (MPW) system parameters, 310
Monitoring processes, 229-233, 234
application performance and, 289-295
batch jobs and, 289-290
interactive jobs and, 290-295

Network management, 333-366
DECnet components and, 340-343
addressing, 340, 341
end nodes and, 341
Level 1 (L1) routers and, 341-343
Level 2 (L2) routers and, 343
DECnet management and, 340, 341, 344-351
assigning addresses and host names and, 344
node configuring, 344-348
shutting down, 348
starting, 348
troubleshooting, 349-351
Ethernet basics and, 333-340
addressing, 334, 335
terminology of, 336-340
Local Area Transport (LAT) and, 340, 351-366
components of, 351, 352-353
configuring terminal server ports and, 359-360
connection basics and, 353
network example of, 352
service nodes (hosts) and, 360-366
terminal servers and, 354-359
Network management listener (NML), 346
Network security, 277-280
DECnet proxy access and, 280
default DECnet access and, 279-280
terminal server passwords and, 278

Object protection and security management, 254-271
access control list (ACL)-based and, 257-263
device, 263, 264, 265, 266
file, 266-271
queue, 271, 272
security class and, 271, 273
user identification code (UIC)-based and, 254-257
volume, 264-266
Objective-based management introduction, 2-3
OPCOM. *See* Operator Console Messenger
Operational site attributes, 13-14
cleaning procedures and, 13-14
definition of, 7
procedures and, 13
Operator Console Messenger, 233-234
Organization of this book, xiv
Owner name, 133

Page file I/O and tuning memory, 309, 310
Page file sizes, establishing, 309, 310-311
PAK. *See* Product Authorization Key

Parameters of the system, managing, 325-332
AUTOGEN utility and, 327-332
SYSMAN PARAMETER command and, 325-326
viewing current values of, 326
Password(s), 133
booting OpenVMS and bypassing, 40-41
content and length, 246-251
custom password filter and, 250-251
Digital-supplied password dictionary and, 250
generated, 249
history data of, 250
learning someone else's, 246
lifetime of, 248
locked, 248-249
minimum length of, 247, 248
multiples of, 247
number of combinations of, 248
privilege password and, 278
security-related system parameters and, 275, 276
server password and, 278
SYSBOOT and bypassing the checking of, 40-41
system, 246-247
terminal server and, 278, 356-357
user accounts and, 133, 135
Performance management, 286-332
application performance and, monitoring, 289-295
batch jobs and, 289-290
interactive jobs and, 290-295
expanding capacity and, 324
introduction to, 286-287
load balancing and, 302-304
CPU and, 304
disk input-output and, 303
memory and, 302-303
objective-based flow chart for, 288
objectives for, identifying, 287, 288-289
samples of, 289
resource dependencies and, identifying, 295-301
estimating CPU and, 298-300
estimating disk input-output and, 297-298
estimating memory and, 295-297
example of, 300-301
sample performance problem and, 324-325
system parameters and, 325-332
AUTOGEN utility and, 327-332
SYSMAN PARAMETER command and, 325-326
tuning CPU and, 321-323
decompress system libraries and, 323
priority boosting and, 322-323
process scheduling and, 321-322
quantum and, 323
tuning I/O and, 316-321
adjust ACP (ancillary control process) caches and, 319-320
adjust volume characteristics and, 318
disk defragmentation and, 316-318
virtual I/O cache and, 320-321
tuning memory and, 305-316
excessive hard page faults and, 305-311
excessive soft page faults and, 311-314
introduction to, 305
reclamation and, 314-316
Physical backups, 190-191
Physical segment and Ethernet, 336
Physical site attributes needed, 8-9
definition of, 7
equipment layout and, 9
flooring and, 9
location and, 8
structural capacity and, 8
Power failures, 9-10

Print forms
creating, 124-125
print jobs using, 125-126
purpose of, 124
Print queues, 104-105
characteristics of, 122
LAT stopping and, 363
load-balancing, 119
names of, default logical, 120
Printer(s)
assigning default, 164
characteristics configuration of, 121, 122-124
configuring for queue management, 112-117
LAT device and, 114-116
printer device and, 116
spooling and, enabling, 116-117
terminal device and, 113-114
Privileges, 390-393
ALL, 392-393
DEVOUR, 391
FILE, 391
GROUP, 390
NORMAL, 390
SYSTEM, 391-392
Proactive memory reclamation (PMR), 315-316
Process scheduling, 321-322
Product Authorization Key, 89-91, 99-102
deleting, 101
electronic representation of, 95-96
enabling and disabling, 100
installing a, 99-100
loading, 100-101
restricting product access and, 101-102
sample of, 90
Purging files, 143-145
file version limits and, 145

QUANTUM, 299, 323
Queue management, 103-126
changes to, summary of, 104
characteristics of, 121, 122
creating and starting, 117-121
Autostart execution of, 117-118
generically, 119-120
standard execution of, 118-119
device control libraries and, 121, 122-124
creating the, 123
creating setup modules for, 121, 122-123
modules to, adding, 123
using, 123-124
introduction to, 103-104
naming conventions of, 111-112
object protection and, 271, 272
print forms and, 124-126
printer devices and, configuring, 112-117
types of, 104-106
batch, 105, 120
print, 104-105, 119, 120
Queue Manager, 106-108
command summary of, 110
controlling, 108-111
database of, 107-108
OpenVMS 6.0 and, 108
DISABLE_AUTOSTART and, 75-76
fail-over and, 109, 110, 120
multiples of, 110-111
restrictions for, 111
post-OpenVMS 5.5, 106, 107, 108-109, 110, 117
pre-OpenVMS 5.5, 106, 107, 108

Reboot, automatic, 72, 73
Resource dependencies, system, 287
estimating CPU and, 298-300
estimating disk input-output and, 297-298
estimating memory and, 295-297
example of, 300-301
identifying, 295-301
Restart handling, 179-181
batch job symbols for, 180
control settings for, 29

Safety equipment site attributes, 12-13
definition of, 7
fire and, 12
power and, 12
security and, 13
water and, 12
Scheduling process, 321-322
Security management, 244-285. *See also* Network security; Object protection and security management; Passwords
access restrictions and, 251-253
ACL-based protection and, 257-263
attributes of, user accounts and, 135, 136
auditing of, 281-285
reports for, generating, 283-285
controlling system access and, 245-253
DECnet access and, default, 279-280
DECnet proxy access and, 280
device protection and, 263, 264, 265, 266
file object protection and, 266-271
introduction to, 244-245
network and, 277-280
new and changed features for, Open VMS 6.0, 245
object protection and, 254-271
password content and length and, 246-251
passwords and, terminal server, 278
protected subsystems and, 272-275
queue object protection and, 271, 272
security class access requirements and, 273
system parameters and, 275-277
UIC-based protection and, 254-257
volume protection and, 264-266
Service-level objectives, developing, 2-3
application requests and, 3
data integrity and, 3
performance and, 3
security and, 3
software upgrades and, 3
system uptime and, 2
user requests and, 2-3
SET PREFIX command, 169, 170
Shadowing, backup and volume, 203-204
Shareable system files, coordinating, 58-60
SHOW SYSTEM, *229*
Shutting down system. *See* System shutdown
Site attributes, 8-14
Site configuration, 7-15
Site guide outline, 369-371
Site-specific startup procedures, 46, 48, 49, 50-53
common-environment VMScluster and, complex, 56-57
common-environment VMScluster and, simple, 55-56
coordinating, 53-58
multiple-environment VMScluster and, 58
stand-alone system and, 55
SYCONFIG.COM and, 50
SYLOGICALS.COM and, 50-51, 59-60
logical names defined in, 51
SYPAGSWPFILES.COM and, 52
SYSECURITY.COM and, 52
SYSTARTUP_Vxx.COM and, 52-53
modularized, 52
templates for, 50
Soft page faults and tuning memory, excessive, 311-314
automatic working set adjustment and, 313-314
establishing working set sizes and, 311-313
detached processes and, 312-313
interactive processes and, 312
Software licenses, Digital, 91-92

Software failure and performing backups, 187
Startup procedures. *See* System startup procedures
Swapper trimming, 315
SYLOGIN, 160-161
SYSBOOT options, 39-42
  adjusting SYSGEN parameters and, 41-42
  bypass password checking and, 40-41
  bypass startup and, 40
  minimal startup and, 39-40
  reverting to DEFAULT parameters and, 42
  reverting to old parameters and, 42
SYSMAN
  defining for login, 172
  disk quota tools and, 148
  shutdown and, 76-77, 81-82
SYSMAN PARAMETER command, 325-326
System configuration, 16-24
  OpenVMS system disk organization and, 16-17
  system logical names and, 18, 19, 20
  VMScluster configurations and, 18-24
System console summary, 372-389
System Dump Analyze (SDA) utility, 231
System libraries, decompressing, 323
System log files, archiving, 157-158
System management. *See* Automating system management
System manager, definition of, 1
System parameters, managing, 325-332
  AUTOGEN utility and, 327-332
  SYSMAN PARAMETER command and, 325-326
System resource dependencies, 287
System shutdown, 68-88
  application type of, 71-72
  automating, 86-87
  default cluster quorum adjustment after, 73
  emergency procedures for, 69, 82-86
    establishing common dump file as an, 85-86
    OPCCRASH as an, 82
    page file as system dump file as an, 85
    performing a crash dump as an, 82-83
    sizing the system dump file as an, 83-85
  introduction to, 68-69
  invoking a, 79-81
  issues for, orderly, 69-76
    automatic reboot and, 72, 73
    notification messages and, 70-71
    site-specific procedures and, 71-72
    spin-down disks and, 71
  notification of, 70-71, 79
  options of, 73-76
    CLUSTER_SHUTDOWN as an, 74-75
    DISABLE_AUTOSTART as an, 75-76
    REBOOT_CHECK as an, 75
    REMOVE_NODE as an, 73-74
    SAVE_FEEDBACK as an, 75
  procedures for, orderly, 76-82
    checking for system activity as an, 77-79
    disable future logins as an, 76-77
    invoking shutdown as an, 79-81
    privileges required to perform an, 69
    sending shutdown notification as an, 79
    SYSMAN shutdown as an, 76-77, 81-82
  situations requiring, 68
  spin-down disks and, 71
  SYSMAN shutdown and, 81-82
    qualifiers for, 81
  typical session for, 79-80

System startup procedures, 45-67. *See also* Booting OpenVMS
batch jobs and, using, 64
booting OpenVMS and bypassing, 40
booting OpenVMS and minimal, 39-40
console logging and, 62-63
dependency issues and, 60-61
error handling and, 47-48, 61
introduction to, 45-46
nonshareable system files and, 60
optimizing the, 61-66
sequence of, 46-50
shareable system files and, coordinating, 58-60
site-independent and, 46, 49
site-specific and, 46, 48, 49, 50-53
coordinating and, 53-58
templates for, 50
snapshot booting and, 54-66
STARTUP phase of, 47-50
SWAPINIT phase of, 46-47
SYSINIT phase of, 47

Temperature range for Digital equipment, 11
Text Processing utility (TPU), 295, 300-301
invoking, 324-325
Tuning
CPU, 321-323
I/O, 316-321
memory, 305-316

Uninterruptible power supply (UPS), 10
User accounts, managing, 127-141
access restrictions of, 136
authorization database for, 127-129
characteristics of, 129-137
example of, 129-130
groupings of, 130
names and, 134
command procedures example of, 140
defaults of, 134-135, 138
deleting, 139, 140
disabling and enabling, 139
identifiers of, 137
introduction to, 127
login attributes of, 135-136
new accounts and, adding, 138-139
owner name and, 133
passwords and, 133
attributes of, 135
privileges of, 137
quotas and limits of, 136-137
reporting, 140
restrictions and, 136
template accounts and, creating, 138
tools of, 140
user identification and, 130-134
username and, 130-131
user identification code (UIC) and, 131-133, 164
User authorization file (UAF)
backups and, performing, 206
configuration for backups and, 206
controlling system access and, 245, 246, 251
login procedure and, 160
User errors and performing backups, 187
User identification code (UIC)-based protection, 254-257
access categories of, 257
object owner and, 255-256
object protection mask and, 256-257
subject owner and, 255
valid formats of, 255

VAXsim Plus (VAX System Integrity Monitor), 227-228
Virtual I/O cache, tuning I/O and, 320-321
VMS system bus types, 372
VMScluster configurations, 18, 19-24
common system disk and, 19, 20
definition of, 18

VMScluster configurations, *continued*
increased visibility and, 21
interconnects and, 21-24
Computer Interconnect (CI) and, 21, 22
Digital Storage System Interconnect (DSSI) and, 21, 22
dual-architecture VMSclusters and, 23
Ethernet and, 22-23
Fiber Distributed Data Interface (FDDI) and, 23, 24
network and, 22-23
remote file access and, 20-21
remote job submission and, 21
service names assignment and, 361
VMScluster shareable data files, 59
Volume characteristics, adjusting, 318
Volume protection, 264, 265, 266, 267
defaults and commands of, 268
Volume shadowing and backups, 203-204

Working set adjustment, automatic, 313-314
Working set sizes, establishing, 311-313
parameters of, 313

OPENVMS SYSTEM MANAGEMENT GUIDE

LAWRENCE BALDWIN

As organizations have downsized computing systems from large mainframes to mini-computers (such as Digital VAX and Alpha systems) and personal computers, there has been an assumption that day-to-day management will also be simplified—allowing systems staff to be similarly reduced.

At first glance, this assumption seems reasonable. Unfortunately, size and shape have little impact on ease of management. OpenVMS is a highly developed and complex operating system, as complex as mainframe-class operating systems. Accordingly, disciplined and methodical systems management practices are essential—not a trivial task to accomplish when systems staffs are shrinking.

The only solution to this dilemma is to manage smarter—automate wherever possible and be proactive rather than reactive. This is the prevailing theme emphasized throughout ***OpenVMS System Management Guide.*** Basic management procedures are not only documented, but also prioritized as to what should be done, and why. Whenever possible, specific procedures are provided to automate or simplify system management tasks.

Lawrence Baldwin is President of System Management Technologies, a consulting firm based in New York, New York. In his over eight years of consulting he has emphasized performance management, VMS systems management, network design and management, systems programming, and VMS internals. His clients have included AT&T, Nynex Mobile Communications, GTE Mobile Communications, and Digital Equipment Corporation.

dp **Digital Press**
An imprint of Butterworth-Heinemann